VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

MOBILNÍ APLIKACE PRO SPOJENÍ LIDÍ NA SPORTOVNÍCH AKCÍCH

MOBILE APPLICATION FOR CONTACTING PEOPLE DURING SPORT EVENTS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **LUKÁŠ JANOUŠEK**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. VLADIMÍR BARTÍK, Ph.D.**
SUPERVISOR

BRNO 2018

Zadání bakalářské práce/21243/2017/xjanou06

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2017/2018

Zadání bakalářské práce

Řešitel: **Janoušek Lukáš**
Obor: Informační technologie
Téma: **Mobilní aplikace pro spojení lidí na sportovních akcích**
Mobile Application for Contacting People during Sport Events
Kategorie: Informační systémy

Pokyny:

1. Seznamte se s principy tvorby mobilních aplikací na platformě iOS a Android.
2. Analyzujte požadavky na aplikaci pro spojení lidí na sportovních akcích. Bude umožňovat zaměření polohy uživatele s tím, že umožní vyhledat nejblíže položené uživatele využívající stejnou aplikaci. Bude také možné plánovat společné akce, soukromé i veřejné.
3. Navrhněte aplikaci splňující výše uvedené požadavky.
4. Navrženou aplikaci implementujte a ověřte její funkčnost na vhodném vzorku dat.
5. Zhodnoťte dosažené výsledky a další možné pokračování tohoto projektu.

Literatura:

- Ujbányai, M.: Programujeme pro Android. Grada Publishing, 2012, ISBN 978-80-247-3995-3.
- Vávrů, J.: iPhone - vývoj aplikací. Grada, 2013. ISBN: 978-80-247-4457-5.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Bartík Vladimír, Ing., Ph.D.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2017
Datum odevzdání: 16. května 2018

L.S.

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Cílem této práce je navrhnout a vyvinout mobilní aplikaci pro platformu Android a iOS. Jedná se tedy o multi-platformní vývoj nativní aplikace psanou scriptovacím jazykem TypeScript. Mobilní aplikace má za úkol spojovat lidi na akcích sportovního charakteru. Každý sportovec má možnost sportovat v kolektivu na místech, které jsou pro samotného sportovce nová a neznámá. Uživatelé mobilní aplikace se mohou připojovat, vyhledávat a vytvářet sportovní akce na určitém místě. Podrobný postup od návrhu až po testování je uveden v textu této bakalářské práce. Výsledkem práce je volně dostupná aplikace pro užší okruh uživatelů.

Abstract

The aim of this thesis is to design and to develop a mobile application for platform of Android and iOS. It is a multi-platform development a native application written in NativeScript language. The mobile application is called to connect people in the events of sportiness character. Every sportsman has a good opportunity to work out in the team on the for a sportsman new and unknown places. Users of mobile applications can join, search and create sports events in a specific location. Detail description of this thesis is designed and tested in this text. The result of the work is a freely available application for a narrower group of users.

Klíčová slova

Mobilní aplikace, Spojování lidí na sportovních akcích, iOS, Android, JavaScript, NativeScript, uživatelské rozhraní

Keywords

Mobile application, Contacting people during sports events, iOS, Android, JavaScript, NativeScript, user interface

Citace

JANOUŠEK, Lukáš. *Mobilní aplikace pro spojení lidí na sportovních akcích*. Brno, 2018. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Vladimír Bartík, Ph.D.

Mobilní aplikace pro spojení lidí na sportovních akcích

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana doktora Vladimíra Bartíka. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Lukáš Janoušek
17. května 2018

Poděkování

Mé poděkování patří zejména mému vedoucímu bakalářské práce panu Ing. Vladimíru Bartíkovi, Ph.D. za odbornou pomoc, připomínky a rady, které mi byly poskytnuty. Další velké poděkování patří mé rodině za podporu a testerům, kteří mi pomohli otestovat danou mobilní aplikaci.

Obsah

1 Úvod **5**

2 Základní prvky systému Android **6**
2.1 Operační systém Android 6
2.2 Historie verzí OS Android 7
2.3 Architektura OS Android 8
2.4 Životní cyklus aktivity 9
2.5 Životní cyklus služby 11
2.6 Podporovaná verze Androidu 13

3 Základní kameny systému iOS **14**
3.1 Operační systém iOS 14
3.2 Historie verzí systému iOS 14
3.3 Architektura operačního systému iOS 15
3.4 Podporovaná verze systému iOS 17

4 Způsob a metoda vývoje mobilní aplikace **18**
4.1 Multi-platformní vývoj aplikace 18
4.2 Nativní vývoj za pomoci JavaScriptu 18
4.3 JavaScript 19
4.4 TypeScript 19
4.5 NativeScript 19
4.6 Framework AngularJS 21

5 Návrh mobilní aplikace **22**
5.1 Popis funkcionality aplikace 22
5.2 Diagram případů užití 22
5.2.1 Role nového uživatele 23
5.2.2 Role stávajícího uživatele 23
5.2.3 Role administrátora 24
5.3 Návrh datového modelu 26

6 Návrh grafického uživatelského rozhraní **28**
6.1 Tvorba rozhraní v NativeScriptu 28
6.1.1 Layouty 28
6.1.2 Ovládací prvky 28
6.1.3 Stylování 29
6.2 Návrh uživatelského rozhraní 29

6.3 Návrh a vytvoření ikony aplikace 31

7 Implementace **32**
7.1 Hlavní bloky aplikace 32
7.2 Služba Firebase 33
7.3 Funkce používání pluginu MapBox 35
7.4 Funkce geolokace za pomoci GPS 36

8 Testování **37**
8.1 Testování za pomoci Emulátoru 37
8.2 Testování aplikace pomocí TestFlight 37

9 Nasazení aplikace do distribuční služby **40**
9.1 Apple Developer Program 40
9.2 Google Play 40

10 Závěr **42**

Literatura **43**

Přílohy **45**

A Obsah přiloženého paměťového média **46**

B Výsledný vzhled mobilní aplikace **47**

Seznam obrázků

2.1 Schéma architektury systému Android 8
2.2 Životní cyklus aktivity 10
2.3 Životní cyklus služby 12
2.4 Podíl zařízení s verzemi Androidu. Stav k dubnu 2018 13

3.1 Vrstvy operačního systému iOS 16
3.2 iPhone 5S 17
3.3 iPhone X 17

4.1 Funkce NativeScriptu 20
4.2 Model-View-controller 21

5.1 Nastavení rozpětí vyhledávání uživatelů 24
5.2 Diagram případů užití 25
5.3 ER diagram vztahů databáze mobilní aplikace 27

6.1 Návrh přihlášení uživatele 29
6.2 Návrh registrace uživatele 29
6.3 Návrh typů událostí 30
6.4 Návrh vytvoření události 30
6.5 Návrh detailu událostí 30
6.6 Návrh mapy událostí s pozicí 30
6.7 Velikost ikony a její rozlišení pro Android 31

7.1 Naplnění databáze Firebase daty 33
7.2 Metoda vložení dat uživatelů do databáze 34

B.1 Nastavení uživatele 47
B.2 Nastavení profilu testera 47
B.3 Určení data a času události 48
B.4 Moje vytvořené události 48
B.5 Vytvořená událost-část 1 49
B.6 Vytvořená událost-část 2 49
B.7 Bod vytvoření události 50
B.8 Detail připojených událostí 50

Seznam tabulek

2.1 Seznam verzí operačního systému Android 8

3.1 Seznam verzí operačního systému iOS 15

8.1 Uživatelské beta testování 39

Kapitola 1

Úvod

V dnešní době vyspělé technické společnosti, kde chytrý mobilní telefon představuje nedílnou součást denní potřeby člověka, získává na vážnosti samotná mobilní aplikace společně s vývojem mobilních aplikací. Uživatelé chytrých mobilních zařízení se zařadili do proudu, kde přicházejí individuální požadavky a tedy individuální vývoj samotných aplikací. Potřeba či nápad je prvním impulzem, jak pro běžného uživatele, tak pro samotné vývojáře aplikací. Je to určitým typem výzvy pro programátory jako takové.

V této bakalářské práci se budu zabývat vývojem nativní mobilní aplikace za užití dvou platforem. Nejpoužívanější platformy dnešní doby jsou operační systémy Android od společnosti Google a iOS operační systém společnosti Apple. Jedná se tedy o multi-platformní vývoj mobilní aplikace. Tyto dvě platformy pokrývají téměř 99% trhu. Zbylé pouhopouhé jedno procento systémů, jako je například Windows Mobile/Phone, je zanedbatelnou součástí trhu a postupně jsou z oblasti trhu vytlačovány.

Mobilní aplikace je psána a vyvíjena skriptovacím jazykem TypeScript za pomoci NativeScriptu, které jsou určitou nadstavbou jazyka JavaScript. Aplikace vyvinuta ve vývojovém prostředí Angular IDE. Mobilní aplikace pro spojení lidí na sportovních akcích je krok za krokem popsána od návrhu, implementace, testování až po nasazení do distribučních služeb daných platforem. Vývoj mobilní aplikace je popsán způsobem, kterým eliminuje prvotní složitost a rozsáhlost daného odvětví, aby složitost nebyla překážkou pro vývoj aplikace.

Kapitola 2

Základní prvky systému Android

V této kapitole popíši stručně historii operačního systému Android spolu se základními kameny Android aplikace a samotného operačního systému. Tato platforma je jednou ze dvou užitých platforem, která byla použita při vývoji mobilní aplikace.

2.1 Operační systém Android

Operační systém Android je multi-platformní platformou typu open–source na bázi Linuxu. V současnosti hlavně užívaný v mobilních telefonech, tabletech, navigacích a dalších zařízeních. Dnes už je běžnou záležitostí i užívání platformy v chytrých televizních přijímačích. Systém Android je vyvíjen organizací *Open Handset Alliance* spadající pod americkou společnost Google.

Daný operační systém podporuje, jako jeden z mála, více platforem různých značek např. Samsung či HTC a mnoho dalších. Tato vlastnost systému se může na první pohled jevit, jako silná výhoda oproti konkurenci, ale nemusí to platit ve všech směrech. Nevýhoda spočívá v optimalizaci samotného systému pro konkrétní platformu. V tomto charakteru dominuje platforma iOS od společnosti Apple. Multi-platforma Android systému s možností nadstaveb a přizpůsobení jako je HTC sense nebo Samsung TouchWiz. Aktualizace systému pro tato zařízení vydané společností Google jsou problémem, protože nejsou ihned dostupná všem typům zařízení pro každého uživatele. Každý výrobce si udělá vlastní úpravu aktualizace sám pro svá zařízení a poté je dostupná všem uživatelům platformy.

Otevřenost platformy a možnosti úprav ze strany výrobců či samotných uživatelů je výhodou i nevýhodou. Úpravy se netýkají jen konfigurace či widgetů, ale i firmwaru. Aplikace na platformě Android mají největší zastoupení na trhu. Velkou část tvoří aplikace, které nedosahují potřebné kvality. Je to způsobeno schvalovacím procesem a podmínkami, které jsou nastoleny. Podmínky schvalovacího procesu u platformy iOS jsou znatelně přísnější než u zmiňované platformy Android.

Chytré telefony s platformou Android mají větší zastoupení na trhu oproti iOS od společnosti Apple. To znamená, že při vývoji aplikací má větší hnací sílu i u nových zařízení oproti největší konkurenci, kterou je iOS. Je to velký rozdíl nejen při vývoji hardwaru pro určitý a jedinečný druh zařízení, který má v rukou jedna společnost jako je Apple. Způsobuje to problém s během aplikací na různých zařízeních s různým výkonem procesorů a grafických karet s rozlišením různých displejů. To pocítíme na komfortu uživatelského rozhraní, které je velmi rozdílné.

Verze operačního systému Android vydávají výrobci zařízení, a tak je důsledkem mnoho verzí, se kterými se uživatel i vývojář jistě setkal a setkávat bude.[13]

2.2 Historie verzí OS Android

Společnost Android vznikla v roce 2003 v kalifornském Palo Alto. Dva roky nato ji odkoupila společnost Google asi za 50 milionů dolarů. V roce 2007 byla založena Open Handset Alliance stojící za vývojem do dnešních dnů. První verze sytému Android 1.0 byla aplikována společností HTC do mobilního zařízení. V roce 2009 byla masově nasazena do mobilních zařízení verze Android 1.5 (Cupkake).[3]

Verzí operačního systému, pro tuto platformu, vyšlo již nespočet. Vůbec první a nekomerční verze systému vyšla roku 2007 s názvem Android Alpha. Poslední vydanou komerční verzí je Android 8.0 Oreo. Verze těchto systémů jsou označovány, jak číselným označením, tak kódovým označením. Kódovým označením je název zákusku a jsou řazeny dle abecedy. To můžeme vidět v následující tabulce 2.1.

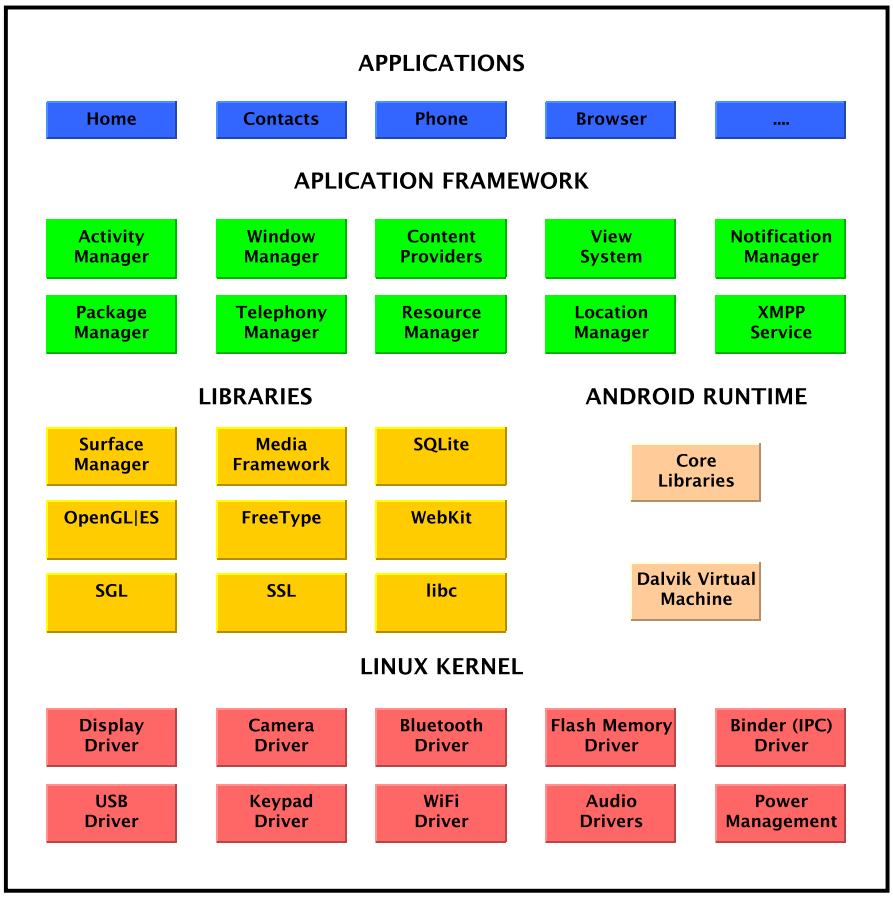
Verze platforem operačního systému Android			
Verze OS	**Jméno**	**Rok nasazení**	**Verze API**
Android 1.0	—	2008	1
Android 1.1	—	2009	2
Android 1.5	Cupcake	2009	3
Android 1.6	Donut	2009	4
Android 2.0	Eclair	2009	5
Android 2.0.1	Eclair	2009	6
Android 2.1	Eclair	2010	7
Android 2.2–2.2.3	Froyo	2010/2011	8
Android 2.3–2.3.2	Gingerbread	2010/2011	9
Android 2.3.3–2.3.7	Gingerbread	2011	10
Android 3.0	Honeycomb	2011	11
Android 3.1	Honeycomb	2011	12
Android 3.2	Honeycomb	2012	13
Android 4.0–4.0.2	IceCream Sandwich	2012	14
Android 4.0.3–4.0.4	IceCream Sandwich	2012	15
Android 4.1	Jelly Bean	2012	16
Android 4.2	Jelly Bean	2012	17
Android 4.3	Jelly Bean	2013	18
Android 4.4	KitKat	2013	19
Android 4.4	KitKat Wear	2014	20
Android 5.0	Lollipop	2014	21
Android 5.1.1	Lollipop	2015	22
(pokračování na další stránce)			

(pokračování seznamu)			
Verze OS	**Jméno**	**Rok nasazení**	**Verze API**
Android 6.0	Marshmallow	2016	23
Android 7.0	Nougat	2016	24
Android 7.1	Nougat	2016	25
Android 8.0	Oreo	2017	26
Android 8.1	Oreo	2018	27
(Konec seznamu)			

Tabulka 2.1: Seznam verzí operačního systému Android

2.3 Architektura OS Android

Pro vývoj mobilní aplikace je nutné mít přehled a základní znalosti architektury systému Android.[10] Architektura je členěna do pěti vrstev. Každá vrstva vystupuje samostatně a provádí určité úkony či operace. V praxi, když je třeba, spolu mezi sebou vrstvy spolupracují. K lepšímu pochopení a představě nám poslouží obrázek 2.1.



Obrázek 2.1: Schéma architektury systému Android

2.4 Životní cyklus aktivity

Definujeme ho metodami, které jsou spouštěny přesně danými situacemi v přesně určenou dobu. Životní cyklus vyjadřujeme nejlépe diagramem, který je vyobrazen na obr. 2.2. Aplikace se skládá z aktivit, které jsou mezi sebou vázány. Aktivity jsou uchovávány v zásobníku typu **LIFO**, která je klasickou frontou. Když uživatel použije tlačítko zpět, tak se vytáhne ze zásobníku předchozí aktivita. Hlavní myšlenkou systému je zachovat mechanismy pro zdroje, jako je baterie nebo paměť. Tyto mechanismy jsou patrné v životním cyklu aktivity. Samotný cyklus definuje určité stavy a události, kterými aktivita celkově prochází od počátku vzniku až po její dokončení.[13] V životním cyklu aktivity lze monitorovat:

1. Úplný životní cyklus aktivity
2. Viditelný životní cyklus aktivity
3. Životní cyklus aktivity v popředí

Úplný cyklus

Cyklus probíhá mezi voláním metody *onCreate*, která aktivity vytváří a metody *onDestroy*, která aktivity ruší. Z důvodu obnovení si aktivita nastavuje globální stav a při ukončování se uvolní všechny využívající prostředky té doby.

Viditelný cyklus

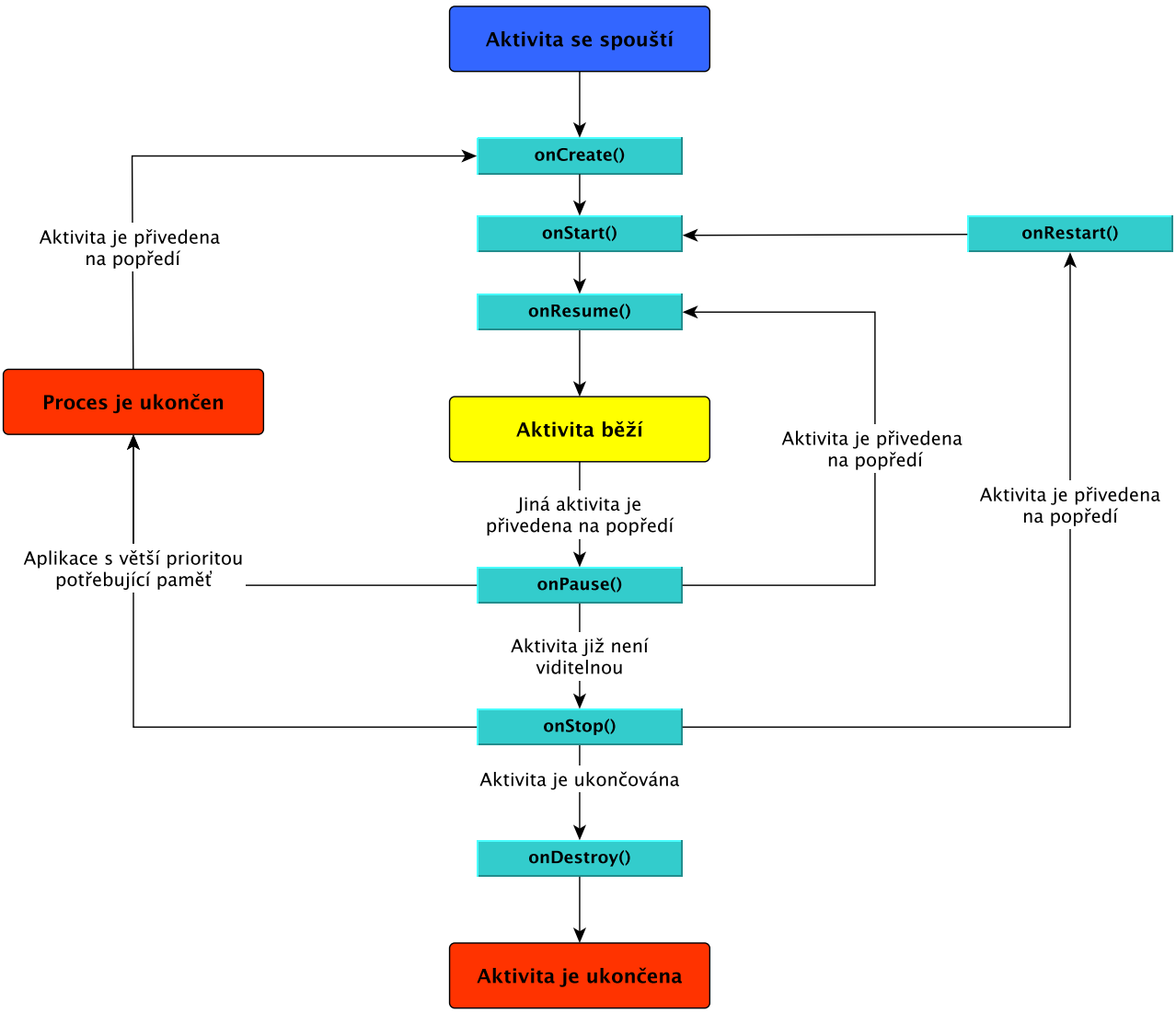
Tento cyklus probíhá ve fázi viditelnosti mezi voláním metody *onStart*, která aktivity spouští a metody *onStop*, která aktivity zastavuje. Uživatel v této době vidí aktivitu na displeji. Prostředky nutné k zobrazení aktivity můžeme zachovat mezi voláním metody *onStart* a *onStop*.

Cyklus v popředí

Tento cyklus nastává voláním metody *onResume*, která umožňuje aktivitě být v činnosti a metodou *onPause* tato aktivita končí. S uživatelem v této době je schopna komunikovat a je zobrazena nad všemi aktivitami té doby. Aktivita se v cyklu může zastavovat a pokračovat několikrát po sobě.

Aktivita reaguje na tyto metody:

- onCreate
- onStart
- onStop
- onPause
- onResume
- onDestroy
- onRestart



Obrázek 2.2: Životní cyklus aktivity

2.5 Životní cyklus služby

Nemalou pozornost a význam v provozovaných službách si zaslouží i životní cyklus služby.[10] Běžný uživatel nemá žádné tušení o službách běžících na pozadí. Musíme tedy věnovat velkou pozornost životnímu cyklu, který je vyobrazen na obrázku 2.3. Ten je velmi podobný životnímu cyklu aktivity z předchozí kapitoly 2.4. Služby dělíme na dva typy:

1. Službu typu Started
2. Službu typu Bound

Služba Started

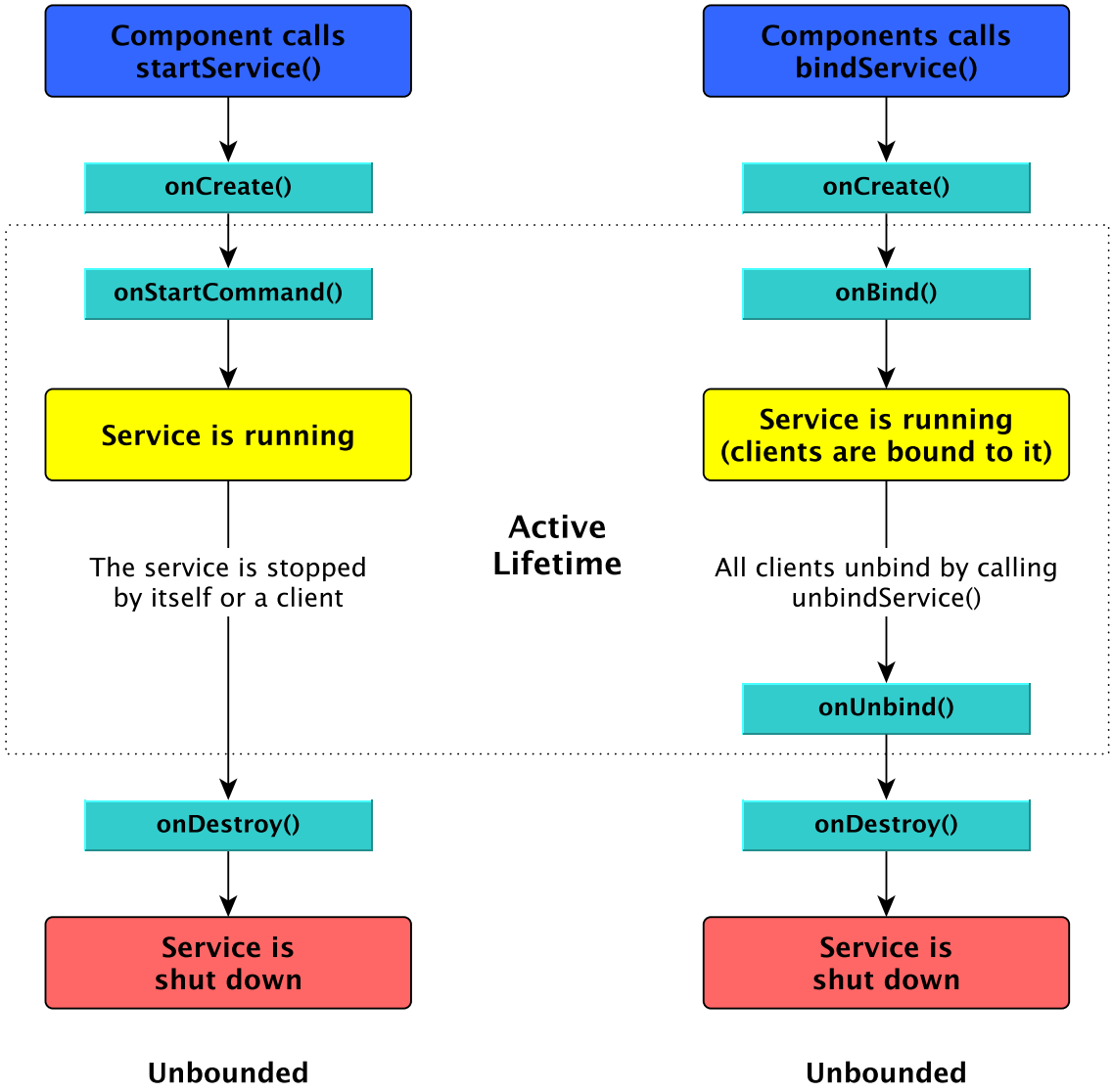
Je to služba, která je volána komponentou aplikace. Toto volání představuje spuštění metody *startService()*. Po spuštění služby na pozadí může služba běžet neomezený čas. Služba běží na pozadí i za podmínky zrušení komponenty aplikace. Služba provede jednu operaci bez návratové informace o výsledku provedení. První možností ukončení činnosti je zprávou jednou z komponent voláním metody *stopService()*. Druhá varianta ukončení je za pomoci volání metody *stopSelf()*.

Služba Bound

Služba tohoto typu se spouští podobným způsobem, jako služba **Started**. Liší se volanou metodou, kde se užívá metoda *bindService()*. Klientům umožňuje služba navázat stálé spojení, komunikovat formou zaslání žádosti a přijetí výsledků konání. Při volbě metody *unbindService()* se klient může odpojit, pokud nadále nebude využívat tuto službu. Existuje-li jedno nebo více stálých připojení, tak je služba stále aktivní. Metodou *onDestroy()* se ruší celá služba **Bound**, kde již není připojen žádný klient.

Služba reaguje na tyto metody:

- onCreate
- onDestroy
- onBind
- onUnbind
- onStartCommand



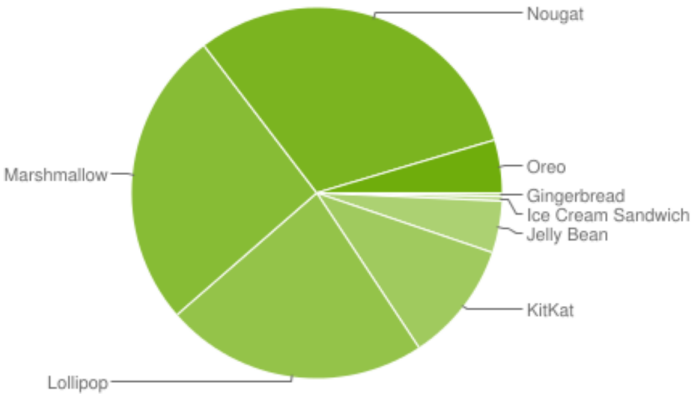
Obrázek 2.3: Životní cyklus služby

2.6 Podporovaná verze Androidu

Nyní je velmi důležité položit si otázku: „Jaké verze systému Android podporovat?“ Životnost mobilního zařízení je v rozmezí dvou až třech let. To už nám samo o sobě napovídá, že systémů se starším operačním systémem razantně ubývá.

Poslední verze systému je dnes Android 8.1 Oreo. Ve vývoji budeme podporovat, jako poslední verzi operačního systému, Android 5.0 Lollipop až po současný systém Android 8.1 Oreo. To vše je možné vidět a porovnat na obrázku 2.4, kde je patrné, že starší systémy už zastupují malou část na trhu s užívanými operačními systémy.[7]

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.3%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.4%
4.1.x	Jelly Bean	16	1.7%
4.2.x		17	2.2%
4.3		18	0.6%
4.4	KitKat	19	10.5%
5.0	Lollipop	21	4.9%
5.1		22	18.0%
6.0	Marshmallow	23	26.0%
7.0	Nougat	24	23.0%
7.1		25	7.8%
8.0	Oreo	26	4.1%
8.1		27	0.5%



Obrázek 2.4: Podíl zařízení s verzemi Androidu. Stav k dubnu 2018

Kapitola 3

Základní kameny systému iOS

Druhou použitou platformou, při vývoji mobilní aplikace, je platforma operačního systému iOS vytvořenou společností Apple. Tato platforma zastupuje na trhu, u mobilních zařízení, druhou nejpoužívanější ihned za operačním systémem Android.

3.1 Operační systém iOS

Operační systém byl původně vytvořen pouze pro mobilní telefony iPhone. Později se systém dostal do zařízení, jako je multimediální počítač iPad typu tablet nebo multimediální přehrávač iPod.

Samotný operační systém je odlehčenou verzí operačního systému macOS, užívaného v počítačích Apple. Mobilní verze má podporu dotykového ovládání. Systém je typu **UNIX**. Celý systém je navržen jako uzavřený systém, což zvyšuje nároky na samotné vývojáře, testování i samotné nasazování mobilních aplikací. Aplikace třetích stran není možné dle potřeby instalovat a používat. K tomu slouží oficiální úložiště aplikací App Store.

Vývoj a testování aplikací pro tuto platformu má výhodu běhu aplikace na jednom typu zařízení od jednoho výrobce. Tím nastávají další plusy systému. Jsou odladěnější, optimalizovanější než jeho konkurence.[14]

3.2 Historie verzí systému iOS

Tuto americkou Společnost Apple založili tři muži v roce 1976 pod názvem Apple Computer. Byli to Steven Paul Jobs, Stephen Wozniak, Ronald Wayne.

Nás nejvíce zajímá vývoj operačního systému a verzí pro mobilní, chytrý telefon iPhone, na kterém poběží naše mobilní aplikace. Verze operačních systémů iOS naleznete v tab. 3.1.

Verze platforem operačního systému iOS			
Verze OS	**Jméno**	**Rok nasazení**	**Nejvyšší dostupný pro**
1.x	iPhone OS	2007/2008	iPhone
2.x	iPhone OS	2008/2009	iPhone 3G
3.x	iPhone OS	2009/2010	iPhone 3GS
(pokračování na další stránce)			

(pokračování seznamu)			
Verze OS	**Jméno**	**Rok nasazení**	**Nejvyšší dostupný pro**
4.x	iOS	2010/2011	iPhone 4
5.x	iOS	2011/2012	iPhone 4S
6.x	iOS	2012/2014	iPhone 5
7.x	iOS	2013/2014	iPhone 5S/5C
8.x	iOS	2014/2015	iPhone 6/6 Plus
9.x	iOS	2015/2016	iPhone 6/6 Plus
10.x	iOS	2016	iPhone 7/7 Plus/SE
11.x	iOS	2017	iPhone 8/8 Plus/X
			(Konec seznamu)

Tabulka 3.1: Seznam verzí operačního systému iOS

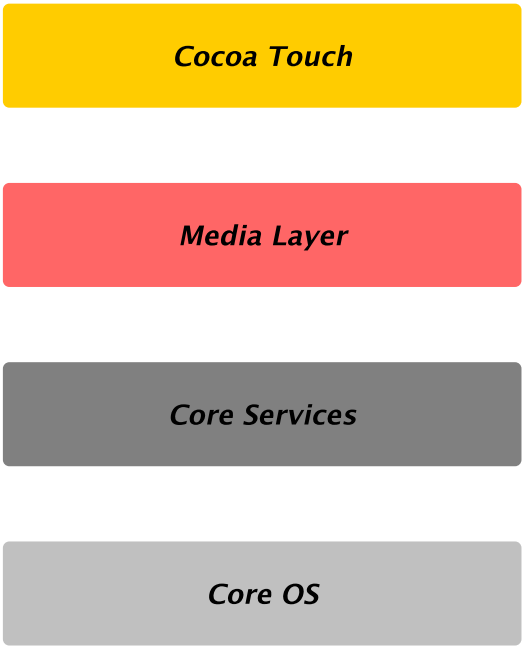
3.3 Architektura operačního systému iOS

Architektura operačního systému je vrstvená. Samotné vrstvy používají vývojáři při vývoji operačních systémů pro rozložení složitosti komplexních systémů. Vrstvy rozdělujeme na úrovně. Toto rozdělení je důležité z pohledu závislosti každé jednotlivé vrstvy. Vždy platí předpis, který udává, že vyšší vrstva užívá služeb vrstev nižších avšak pouze o jednu úroveň nižší. Nižší vrstva neví nic o vrstvách vyšších. Nižší vrstva nabízí pouze rozhraní známé pouze vyšší vrstvě. To je hlavním principem vrstvení této architektury.

Velice významnou výhodou vrstvení je zjednodušení složitějších systémů nebo efektivnější testování. To spočívá v pochopení pravidel použití vrstvy, kterou chceme použít a tím využijeme celý složitý podsystém. Efektivnější testování spočívá v rozdělení složitější funkcionality do menších vrstev, a tak pokryjeme vetší funkcionalitu testy.

Naopak to přináší i nevýhody, kterou je např. snížení výkonu systému. Při přidání každé další vrstvy může dojít k celkovému snížení výkonu. Každá vrstva má vlastní reprezentaci entit a při předávání entit mezi jednotlivými vrstvami dochází ke transformaci. Každá transformace potřebuje výkon a tím je systém následně zpomalen.

Operační systém iOS na nejvyšší úrovni pracuje, jako prostředník, mezi základním hardwarem a aplikacemi, které uživatel provádí. Samotná aplikace přímo nekomunikuje s nejnižší hardwarovou vrstvou. Aplikace s ní komunikuje za pomocí sady definovaných systémových rozhraní. Základní vrstvy operačního systému iOS jsou čtyři na obr.3.1.



Obrázek 3.1: Vrstvy operačního systému iOS

Vrstva Core OS

Zde v hlavní vrstvě jsou zachovány funkce nízké úrovně, které jsou např. Core Bluetooth Framework nebo Security Services Framework.

Vrstva Core Services

Tato vrstva zajišťuje obsluhu služeb, které zajišťují např. programový přístup do databáze kontaktů uživatele a nazýváme ho Address book framework. Služba Cloud Kit framework poskytuje médium pro přesun dat mezi naší mobilní aplikací a servrovým úložištěm iClaud.

Vrstva Media Layer

Tato vrstva slouží pro ovládání grafického rámce, audia a videa. Umožňuje plynulé přehrávání audia, videí i animací. Grafický rámec UIKit Graphics popisuje vysokou úroveň pro vyvíjení obrázků a poskytuje užití animací obsahu uživatelských pohledů.

Vrstva Cocoa Touch

Tato vrstva pracuje jako finální vrstva z pohledu designového pro celou aplikaci. Vlastně poskytují infrastrukturu pro implementaci grafického rozhraní aplikace a interakci s uživatelem. V této vrstvě nalezneme GameKit Framework, který implementuje podporu pro herní centrum, kde uživatelé sdílí informace o svých hrách.

3.4 Podporovaná verze systému iOS

Možnost srovnání užívání verzí operačního systému iOS, tak jak je vyobrazen na obr. 2.4, kde je podíl jednotlivých verzí operačního systému Android, není u systému iOS nutný. Do mobilních zařízení iPhone je aktualizovaná nejvyšší podporovaná verze operačního systému. Posledním podporovaným zařízením je iPhone 5 a iPhone C, pro které již skončila podpora ze strany vývojářů společnosti Apple.

Vyvinutá mobilní aplikace u operačního systému iOS podporuje verzi iOS 8 a vyšší. Tato verze podporuje modelovou řadu mobilních telefonů, společnosti Apple, od iPhone 5S a vyšší modelové řady. Přehled verzí OS je v tabulce 3.1.

Na těchto obrázcích 3.2 a 3.3 můžete vidět dvě modelové řady iPhone, pro kterou je daná aplikace vyvinuta. Od nejstaršího podporovaného modelu 5S po nejnovější model iPhone X s operačním systémem iOS 11.

Obrázek 3.2: iPhone 5S

Obrázek 3.3: iPhone X

Kapitola 4

Způsob a metoda vývoje mobilní aplikace

Mobilní aplikace se v dnešní době vyvíjí mnoha způsoby pro různé platformy. Vývojář má možnost výběru. Vyvinout aplikaci jen pro vybranou platformu, kterou si zvolí, kde se většinou rozhoduje mezi Androidem nebo iOS systémem. Další možností je vyvíjet aplikaci multi-platformním způsobem sestaveným za pomoci JavaScriptu, který jsem si také zvolil. Zvolil jsem tak z důvodu většího pokrytí budoucích uživatelů mobilní aplikace. Dle mého názoru se nevyplatí zabívat se vývojem jedné vybrané platformy. Zde si popíšeme použité technologie aplikace, které byly během vývoje zvoleny a použity.

4.1 Multi-platformní vývoj aplikace

Je to praxe, při které se vyvíjejí softwarové produkty pro větší množství platforem za použití metody přizpůsobení aplikace více operačním systémům.

Největším plusem tohoto způsobu vývoje je možnost získání velkého počtu uživatelů a tím i pokrytí dvou nejrozšířenějších operačních systémů. Výsledkem je pro vývojáře lepší zaměření marketingu při nabízení aplikace v distribučních službách.[11]

Přístup multi-platformního vývoje k problematice je vytvořit několik vývojových větví. Každá platforma bude mít svoji větev. V praxi to má dopad na vývoj unikátního kódu pro každou z platforem. To má za následek obtížnější udržitelnost aplikace. Vývoj a případné opravování chyb je dražší, protože je znatelná časová náročnost. Hlavní myšlenka metody je vytvoření dvou různých programů se schopností velmi podobného chování. Možný další přístup je abstrakce platformy, který je závislý na softwaru skrývající rozdíly mezi platformami. V případě vývoje pro dvě platformy je lepší jíti směrem jednotného kódu, který bude přehlednější a lépe udržitelný. Výsledkem je jedna aplikace, kde je nízká cena nákladů na vývoj i čas.[11]

4.2 Nativní vývoj za pomoci JavaScriptu

Vývoj nativní aplikace se zaměřuje pro konkrétní platformu/mobilní zařízení. Pro operační systém Android se aplikace vyvíjí v jazyce Java nebo C# za použití vývojového prostředí, kterým je Android Studio. Pro operační systém iOS se vyvíjí aplikace v jazyku Swift nebo Objective-C za použití vývojového studia Xcode. Samotná nativní aplikace je za běhu velmi

rychlá, protože je vyvíjena přímo proti nativnímu API a nativní UI prvky. Nevýhodou nativního způsobu tvorby aplikací je samotná absence multi-platformního vývoje.

Způsob vývoje za pomoci JavaScriptu se dostal do podvědomí vývojářů a získává tak na důležitosti díky uvolněným frameworkům typu Angular 2, React Native a NativeScript. Tyto aplikace sdílí programovací jazyk pro obě vyvíjené platformy. Ovládací komponenty aplikace vytvořené technologií HTML a JavaScript, které by byly vloženy do komponenty WebView tu nejsou. Ovládací prvky jsou tu řešeny způsobem nativních ovladacích prvků daného operačního systému.

Použití nativního vývoje pomocí JavaScriptu je interpretace kódu za běhu, kde není nutná kompilace samotného kódu. Tato technologie používá vlákno, které JavaScript zpracovává. Důvodem je volání funkce nativních API vytvářející uživatelské rozhraní dané aplikace.

Velkou výhodou tohoto způsobu je právě multi-platformní vývoj aplikaci. Vývoj je tak časově nenáročný i s ohledem na cenu vývoje. Jednoduchost testování je tu výhodou pro obě platformy aplikace.[12]

4.3 JavaScript

JavaScript je multi-platformním, objektově orientovaným skriptovacím jazykem, který stvořil Brendan Eich. Skriptovací jazyk je programovací jazyk, navržený pro snadné zvládnutí, rychlost a jednoduchost při vývoji programů.[8]

JavaScript sloužil pro interakci s prvky uživatelského rohraní webových aplikací. V dnešní době, za pomoci JavaScriptu vyvíjíme Single-page aplikace a je možné ho využít též pro backendovou část webových aplikaci. V případě mobilního vývoje slouží pro komunikaci s nativní částí mobilní aplikace. V mém případě vývoje mobilní aplikace se jedná o NativeScript.

4.4 TypeScript

TypeScript, který jsem používal je open-source programovací jazyk, vytvořený a spravovaný firmou Microsoft. Jedná se o určitou nadstavbu nad jazykem JavaScript, která jej rozšiřuje o statické typování a další atributy, které známe z objektově orientovaného programování (moduly, třídy, rozhraní..).[5]

4.5 NativeScript

Technologie NativeScript je rámec (framework) typu open-source pro vývoj aplikací. Aplikace je vyvinutá NativeScriptem založená a postavená pomocí JavaScriptu. Nemusí to být jen JavaScript, ale jazyk, který do JavaScriptu přechází. Vyvíjená aplikace postavená na této metodě vede čistě k nativní aplikaci. Ta využívá stejné API, jako by byla vyvinuta zvláště v Xcode prostředí nebo Android studiu. Z toho plyne výhoda této metody multi-platformního vývoje.[4]

Funkce NativeScriptu je postavena na čtyřech hlavních částech. Hlavní části jsou vyobrazeny na obr.4.1, kde slouží k lepšímu pochopení funkce NativeScriptu.[2] Tyto části jsou:

1. Runtimes

2. Core Modules
3. NativeScript CLI
4. NativeScript Plugins

Runtimes

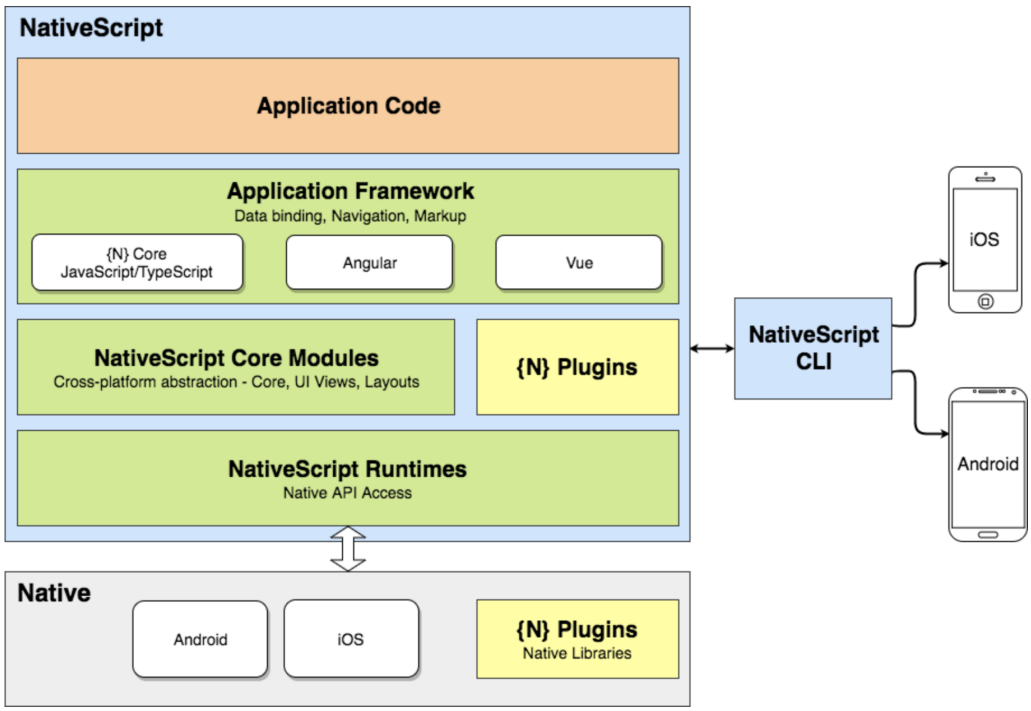
Část Runtimes umožňuje volání rozhraní API v rámci operačních systémů Android a iOS za pomoci kódu JavaScript. Systém Android využívá JavaScript Virtual Machines-Google's V8. Systém iOS využívá WebKit a JavaScriptCore, což je název renderovacího jádra prohlížeče implementovanou s iOS 7.0 a vyšší.

Core Modules

Core Modules jsou tu k dispozici pro poskytnutí abstrakcí potřebných pro přístupu k základní vrstvě nativních platforem. Jako příklad postačí modul gest (The Gestures module). To definuje společné rohraní JS API, který překládá kód aplikace JavaScript do doby volání API díky části Runtimes. Další věcí, co poskytuje Core Modules je XML-based. Způsob definování uživatelského rozhraní, vázání dat a navigaci.

NativeScript Plugins

NativeScript Plugins jsou stavební kameny nebo bloky, které slouží k zapouzdření určité funkcionality. To má za následek rychlejší vývoj aplikace. Většina z nich je psána JavaScriptem nebo TypeScriptem. Některé mohou obsahovat nativní knihovny, které jsou volány z JavaScriptu díky Runtimes.



Obrázek 4.1: Funkce NativeScriptu

NativeScript CLI

Tato část slouží k vytváření a spouštění aplikace za pomoci rozhraní příkazového řádku, které běží ve Windows, Linux a MacOS.

4.6 Framework AngularJS

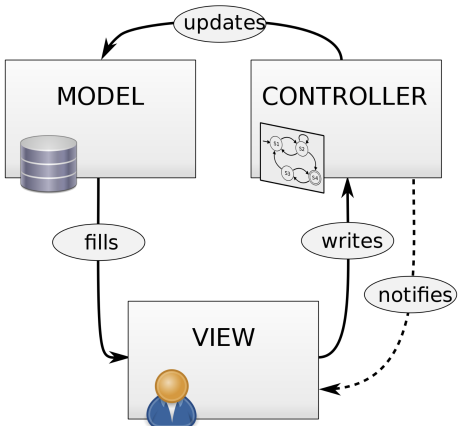
Při vývoji mobilní aplikace jsem využíval služeb JavaScriptového webového aplikačního rámce (framework) AngularJS. AngularJS vytvořený společností Google je volně dostupný na webu. Zaměřuje se na tvorbu single-page aplikací, které jsou tvořeny za pomocí HTML kódu. Kód využívá způsob, který se označuje jako data-binding. Data-binding vytváří formátovací značky určující operace nebo data, které mají být vloženy na předem určené místo.[15]

Hlavní myšlenkou vzniku AngularuJS je pomoc vývojářů stránek a aplikací oddělit logiku zobrazovací od logiky aplikační. Proto AngularJS využívá návrhový vzor Model-View-Controller, díky čemu je aplikační logika v controlleru. Ten je následně vložen do view šablony, která používá definované proměnné a za pomoci direktiv je vypisuje.[1]

Model–view–controller

Model-View-controller je softwarová architektura rozdělující datový model aplikace, řídící logiku a uživatelské rozhraní do třech komponent. Při vývoji aplikace pro iOS je velice vhodné užití tohoto návrhového vzoru. Vytvořená aplikace využívá architekturu MVC obsahující tři části komponent, které jsou na obrázku 4.2. Komponenta Model je specifická reprezentace informací, se kterou pracuje aplikace. View převádí data, která jsou reprezentována modelem do podoby k interaktivní prezentaci uživateli. Controller zpracovává data a nastavuje na View informaci pro zobrazení. Nevýhodou modelu je možnost programátora předat velkou zodpovědnost a tím omezit přehlednost vyvinutého kódu.[6]

1. Model
2. View
3. Controller



Obrázek 4.2: Model-View-controller

Kapitola 5

Návrh mobilní aplikace

Hlavním principem a cílem vývoje tohoto druhu mobilní aplikace je možnost propojení uživatelů na sportovních akcích za použití stejné mobilní aplikace. Dále možnost vytvoření sportovních událostí, na které je možné se připojit. Vyhledávání vytvořených událostí za pomoci mapy a geolokace pomocí GPS. Toto jsou hlavní funkce návrhu této aplikace.

5.1 Popis funkcionality aplikace

Aplikace má za úkol býti uživatelsky velice jednoduchou, aby nebyl sebemenší problém při prvotním použití a s ovládáním novými uživateli.

Uživatelé tohoto druhu aplikace by měli být především samotní aktivní sportovci nebo rekreační nadšenci zapálení do sportování vyhledávající kolektiv a možné seznámení. Pro tuto cílovou skupinu je aplikace zaměřena a vyvíjena. Uživatelé za pomoci této mobilní aplikace získají jednoduchý přístup a přehled o sportovních akcích, které se budou konat v blízké budoucnosti a v určité vzdálenosti od místa výskytu uživatele aplikace. Tyto sportovní akce vytvářejí sami uživatelé mezi sebou. Díky tomuto druhu aplikace může dojít k seznámení mezi účastníky sportovní akce při provozování sportovní činnosti. Je to velice užitečné pro sportovce, kteří nechtějí sportovat sami a hledají vhodné kolegy pro sportování. Sportovci se mohou nacházet v prostředí nebo místech, které neznají.

Novým a neznámým prostředím může být pro člověka či sportovce nové město, které nezná tak dobře nebo je jen na dovolené a hledá sportovní aktivity nebo parťáky. A v tom nastává problém.

Jen těžko může sportovec vědět o plánovaných sportovních aktivitách různého druhu v jeho okolí výskytu, kterých se sám může zůčastnit. Tento problém může vyřešit vývoj nové mobilní aplikace, kterou vyvíjíme. Mobilní aplikaci jsem pojmenoval SportEventsmapp. Samotný název aplikace vypovídá o jejím účelu, poslání a důvodu vývoje.

5.2 Diagram případů užití

Za použití diagramu případu užití, který je na obrázku 5.2, kde je reprezentována funkčnost, hranice a možnosti použití celé mobilní aplikace si popíšeme z pohledu uživatele.

Každý systém založený na typu back–end má různé uživatelské role. V tomto systému jsou užity tři role. Role administrátora, nového uživatele a stávajícího uživatele. Tyto role mají určité další funkce, které si popíšeme ve větším detailu.

5.2.1 Role nového uživatele

Hlavní funkcí role nového uživatele je možnost registrace či vytvoření uživatelského účtu pro přístup k mobilní aplikaci. Tato registrace je nezbytnou součástí pro používání mobilní aplikace. Možnosti registrace jsou dvě. Za pomoci vytvořeného formuláře nebo možnosti registrovat se díky sociální síti Facebook. Při zvolení možnosti registrovat se za pomoci formuláře je nutné vyplnit tyto údaje: jméno, příjmení, uživatelské jméno, e-mail a zvolit si heslo. Pokud uživatel zvolí možnost přihlášení se za pomoci Facebooku, tak se automaticky přihlásí do aplikace. Aplikace si dostupné informace o uživateli sama stáhne za pomoci propojení se sociální sítí Facebook.

5.2.2 Role stávajícího uživatele

Role stávajícího uživatele je nejvíce zastoupený funkcemi ze všech tří základních rolí, které jsou použity. Role a jejich funkce si popíšeme podrobněji, než tomu bylo u zbylích dvou rolí. Popisovat význam rolí a jejich funkce u všech, jako třeba přihlášení či odhlášení je velmi triviální pro pochopení a tedy zbytečné. Všechny obsažené role stávajícího uživatele v diagramu jsou (přihlášení, odhlášení, vytvoření události, zrušení události, prohlížení událostí, nastavení, připojení událostí, vyhledávání událostí, smazání účtu a detail událostí).

Vytvořit událost

Uživatel, který provádí vytváření události zadává do formuláře typ události, datum, čas, telefonní číslo, maximální počet účastníků akce a místo konání události. Místo konání se zadává speciálním způsobem přes zobrazenou mapu, do které uživatel klikne a přidržením prstu na mapě v místě, kde se událost má konat vytvoří bod konání. Nepovinným údajem při vytváření události je podrobný popis události.

Vyhledat událost

Vyhledávání událostí se provádí přes zobrazenou mapu, na které jsou zobrazeny body, které zobrazují vytvořené události, na které se lze přihlásit. Druhým způsobem je dostupný seznam událostí, ke kterým se lze připojit v dostupnosti, kterou určuje nastavené rozmezí uživatelem v nastavení, které je popsáno v kapitole 5.2.2.

Přihlášení

Přihlašování již existujícího uživatele je za pomoci sociání sítě Facebook nebo formuláře, ve kterém zadává uživatelské jméno a heslo, které vytvořil při registraci za pomoci formuláře.

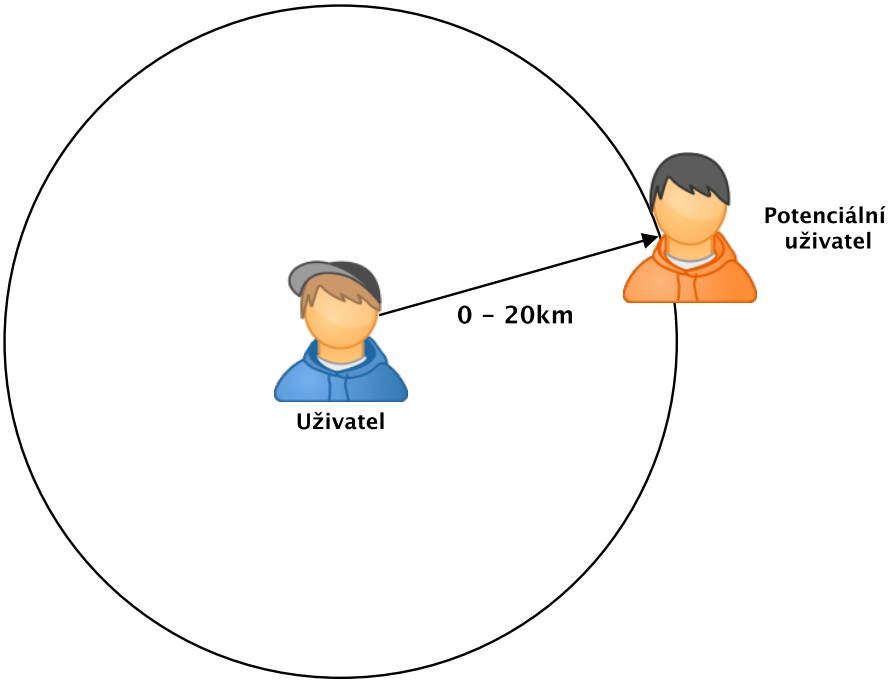
Registrace

Registrace se provádí pomocí Facebooku nebo formuláře. Při registraci přes Facebook je možnost zadat přezdívku uživatele, pokud tuto možnost využil a vlastní účet u této společnosti. Tuto přezdívku je možné zadat pouze jednou.

Nastavení

Uživatel v nastavení aplikace má možnost odhlášení se z aplikace, po kterém bude nutné, při opětovném spuštění aplikace, se opět přihlásit jednou z možností již uvedenou v kapitole 5.2.1. Další možností je nastavení škály pro vyhledávání sportovních událostí od místa

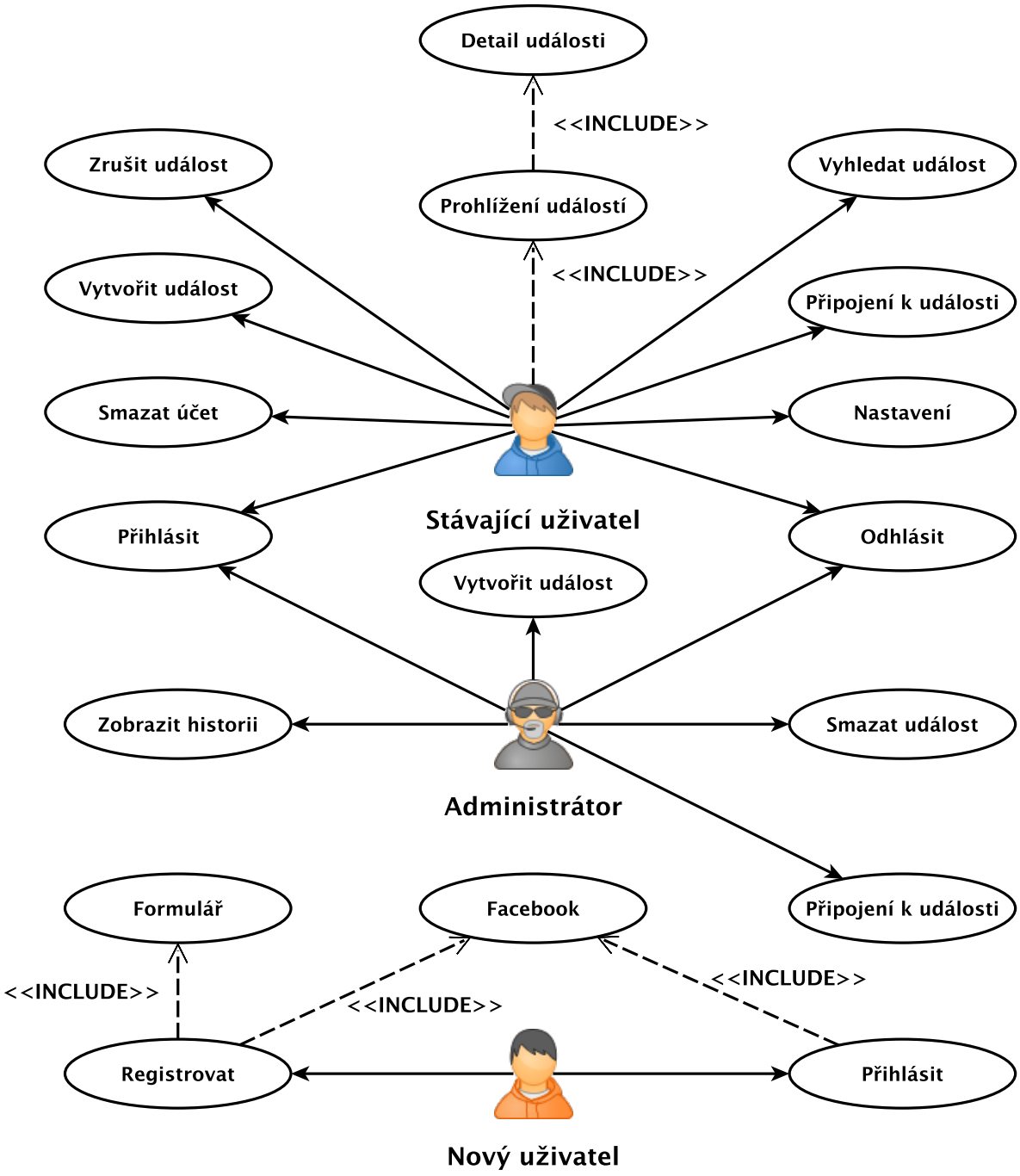
výskytu uživatele aplikace. Na obrázku 5.1 je vyobrazen princip nastavení vyhledávacího rozmezí vzdálenosti. Tuto funkci si může uživatel nastavit dle své potřeby a uvážení. Škála nastavení je od 0 po 20km. Možností nastavení je zadat přezdívku uživatele, pokud využil registraci pomocí sociální sítě Facebook. Tuto přezdívku je možné zadat pouze jednou a to z důvodu znemožnění uživateli měnit přezdívky za použití jednoho vytvořeného účtu. Jako poslední funkcí nastavení je smazání celého účtu uživatelem.



Obrázek 5.1: Nastavení rozpětí vyhledávání uživatelů

5.2.3 Role administrátora

Role administrátora plní funkci prohlížení historie v seznamu událostí, které již proběhly. Dále přihlášení, odhlášení a mazání událostí. Po přihlášení admina unikátním administrátorským účtem a heslem je možné tyto události prohlížet společně s detailem události. Dále nabízí seznam registrovaných uživatelů. Sám administrátor má již zmíněnou možnost odhlášení se, přihášení se z aplikace. Poslední možností je smazání událostí z databáze historie, které již proběhly a nejsou třeba k dalšímu využívání, jako vuužití pro určité statistiky používání aplikace. Proto byl vytvořen list historie pro již konané události.



Obrázek 5.2: Diagram případů užití

5.3 Návrh datového modelu

Pro návrh datového modelu databáze na konceptuální úrovni jsem vybral jednu z metod konceptuálních schémat systému, kterým je ERD (Entity-Relationship Diagram) diagram. Za pomoci ER diagramu, který je na obrázku 5.3 si zobrazíme modelovaná data a jejich vztahy. Tento návrh, popisující uložená data v systému mobilní aplikace, je na úrovni abstrakce. Data a vztahy mezi těmito daty, která potřebujeme uchovávat v systému aplikace si podrobněji popíšeme. Výsledkem návrhu datového modelování je schéma databáze.

Uživatel

Uchovává uživatelovo jméno, příjmení, uživatelské jméno, e-mail, heslo, data o nastavené vzdálenosti. Uživatel má možnost přihlášení a registrace pomocí Facebook účtu. Uživatel vytváří události, maže je, edituje, přihlašuje se a odhlašuje se z událostí.

Události

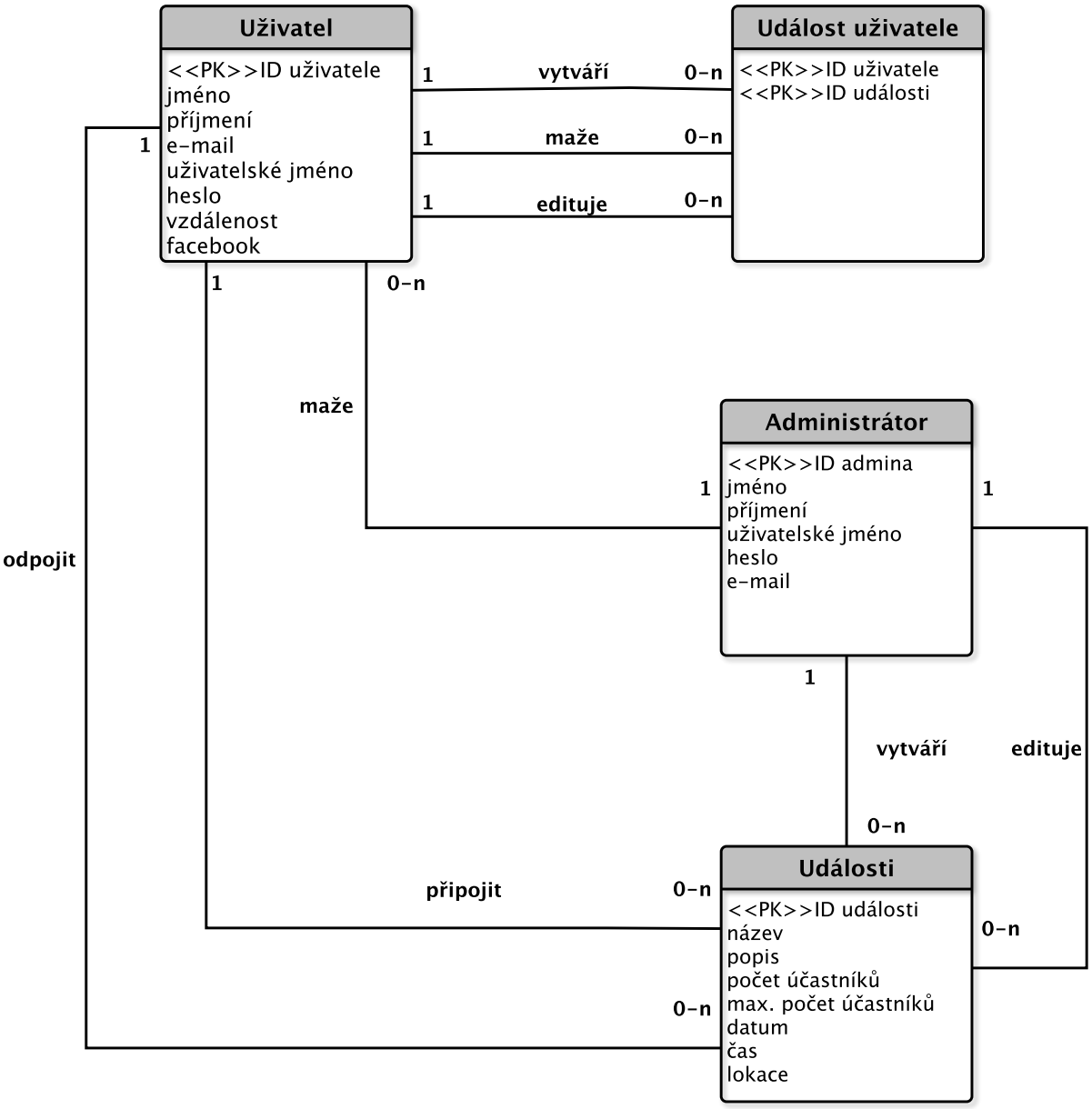
Uchovává se název, popis, počet účastníků, maximální počet účastníků, datum, čas a lokace, která se udává za pomoci bodu na mapě.

Administrátor

Uchovává jméno, příjmení, uživatelské jméno, heslo, e-mail a historii vytvořených událostí. Slouží k mazání účtů, událostí a prohlížení již konaných událostí.

Událost uživatele

Uživatel vytváří, maže a upravuje své vytvořené události.



Obrázek 5.3: ER diagram vztahů databáze mobilní aplikace

Kapitola 6

Návrh grafického uživatelského rozhraní

Grafické uživatelské rozhraní (*Graphical User Interface*) nám umožňuje ovládat mobilní aplikaci a pracovat tak určitým způsobem s elektronickým zařízením. Ovládání je pomocí interaktivních grafických ovládacích prvků.

6.1 Tvorba rozhraní v NativeScriptu

K samotnému vývoji uživatelského rozhraní v NativeScriptu lze přistoupit dvěma různými způsoby. Můžeme pro každou stránku aplikace vytvořit zvláštní soubor XML, kde bude následně definována každá stránka aplikace zvlášť nebo jako jednu stránku s tlačítky, kde každé zobrazí jiná data.

6.1.1 Layouty

NativeScript nám umožňuje grafické rozvržení jednotlivých částí na stránce definovat až šesti způsoby. Každý layout má specifické vlastnosti, které se dají v aplikaci využít.

Například definováním modulu ***scrollView*** umožníme uživateli v určité části stránky skrolovat. Definováním ***stackLayout*** můžeme části stránky uspořádat ve vertikální nebo horizontální poloze. Modul ***gridLayout*** nám umožňuje uspořádat data takovým způsobem, jako bychom je měly v tabulce. Díky ***absoluteLayout*** můžeme data na stránce vypisovat koordináty, které definujeme jako vzdálenosti od levého horního okraje. ***DockLayout*** umožňuje umístit komponenty do stran nebo centra obrazovky. Jako posledním příkladem je ***wrapLayout***, který řadí komponenty uživatelského rozhraní do řádků nebo sloupců. Automaticky jsou řazena horizontálním směrem.

6.1.2 Ovládací prvky

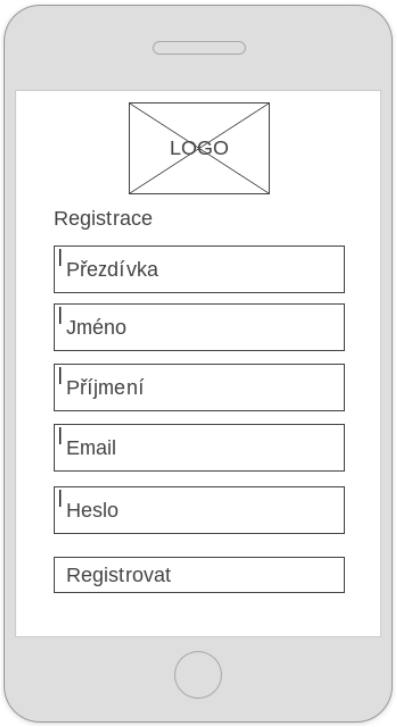
Ovládací prvky, které jsou součástí grafického rozhraní, vyvolají jednu z definovaných akcí. Ovládací prvky jsou v NativeScriptu základními komponenty celé mobilní aplikace. Takovým ovládacím prvkem je **Button**, **Label** nebo **ListView**. Například ListView se při překladu do nativní aplikace zobrazí v Androidu jako *android.widget.ListView* a v iOS jako *UITableView*. NativeScript nám definuje tyto komponenty a následně je přeloží pro každou platformu aplikace tak, aby ji interpretovala stejně.

6.1.3 Stylování

Každý element nebo komponentu můžeme také upravovat pomocí CSS stylů. CSS je jazyk, kterým se definují způsoby zobrazení komponent HTML a XML dokumentech. Umožňuje nám oddělit vzhled dokumentu od jeho struktury a obsahu. Díky NativeScriptu můžeme oddělovat stylování pro každou platformu zvlášť. Tímto způsobem získáváme výhodu kontroly nad zobrazovanými komponentami. NativeScript obsahuje základní témata, která můžeme začít používat ihned po založení nového projektu. Komponenty jsou potom například stylované do jedné určité barvy.

6.2 Návrh uživatelského rozhraní

Jako návrh uživatelského rozhraní mobilní aplikace jsem použil tzv. drátěný model a to za pomoci volně dostupné webové aplikace Wireframe.[1] Jedná se o návrh částí aplikace a definici, kde a jakým způsobem budou komponenty umístěné v mobilní aplikaci. V aplikaci Wireframe si jednoduše můžeme rozvrhnout, jak a kde se budou data zobrazovat, jaká tlačítka bude aplikace obsahovat a způsob jejich navigace. Tlačítka a jejich návrh je možné vyhledat i na webových stránkách Wireframe.cc.[2]



Obrázek 6.1: Návrh přihlášení uživatele

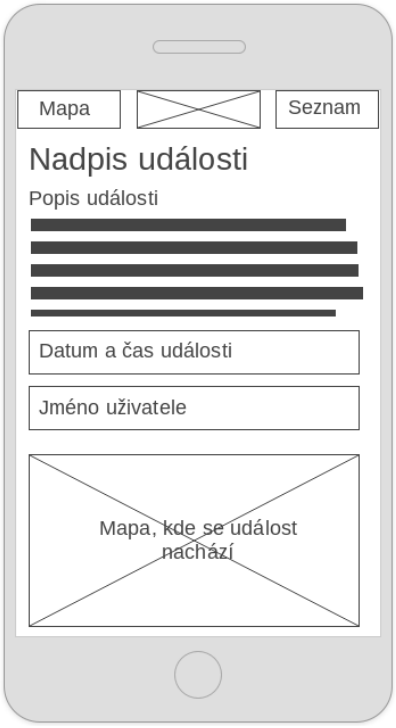
Obrázek 6.2: Návrh registrace uživatele

[1] Webová aplikace Wireframe `https://wireframe.cc/`

[2] Drátěný návrh v aplikaci Wireframe.cc `https://wireframe.cc/pro/pp/d7424c100166428ww63f61m`

Obrázek 6.3: Návrh typů událostí

Obrázek 6.4: Návrh vytvoření události



Obrázek 6.5: Návrh detailu událostí

Obrázek 6.6: Návrh mapy událostí s pozicí

6.3 Návrh a vytvoření ikony aplikace

Ikony a všechny její potřebné rozměry pro obě platformy byly vytvořeny pomocí NativeScript Sidekick. NativeScript Sidekick je grafické uživatelské rozhraní postavené na schopnostech vývojového prostředí Angular IDE, kde je možnost využití určitých typů šablon. Lze přes ni taktéž spravovat všechny pluginy a další nastavení pro obě platformy. Na obrázku 6.7 je znázornění, kde jsou ikony vytvořeny pro platformu operačního systému Android. Pro samotný návrh loga a tlačítek jsem využil Photoshop a volně dostupné ikony.[3]

Obrázek 6.7: Velikost ikony a její rozlišení pro Android

[3]Volně dostupné ikony `http://www.iconarchive.com/`

Kapitola 7

Implementace

Mobilní aplikace je napsaná skriptovacím jazykem TypeScript, který je určitou nadstavbou jazyka JavaScript, za pomoci vývojového prostředí Angular IDE. Kód aplikace se vždy zkompiluje do JavaScriptu. Základní struktura NativeScript projektu je v TypeScript s frameworkem Angular.

7.1 Hlavní bloky aplikace

Hlavní bloky aplikace vytvořené v NativeScriptu a Angularu jsou moduly a komponenty. Aplikace v Angularu jsou modulární. Modulární znamená, že se skládá z modulů, které lze libovolně spojovat do jednoho celku.

Modul

Modul je soubor obsahující blok kódu, který je určený pro jediný účel. Exportuje hodnotu, kterou mohou použít jiné části aplikace. Např:

```
export class AppComponent {}
```

Moduly můžeme následně importovat do jiných modulů. Např:

```
import { AppComponent } from ,./app.component';
```

Komponenty

Komponenty jsou základní bloky aplikace v NativeScriptu s Angularem. Každá aplikace obsahuje komponenty, které definují každou část uživatelského rozhraní nebo routy. Každá komponenta je uvozena dekorátorem @Component, který obsahuje metadata popisující, jak budou komponenty zobrazeny a vytvořeny.

Životní cyklus komponenty je kontrolován Angularem 2. Stará se o vytváření, obnovu a zánik komponent. Například v mé aplikaci je využita funkce *ngOnInit()*, která je volána ihned po inicializaci všech vstupních dat a to pouze jednou před funkcí *ngOnChanges()* a po volání konstruktoru. V mé aplikaci provádím v *ngOnInit()* inicializaci proměnných typu Observer. Observer je proud událostí, které můžeme sledovat, kde Observer je ten, který sleduje.

```
ngOnInit() {
    this.token = BackendService.token;
```

```
    this.events$ = <any>this.firebaseService.getMyEventList();
    this.assigns$ = <any>this.firebaseService.assignEventList();
    this.allevents$ = <any>this.firebaseService.getEventList();
  }
```

Tyto proměnné jsou v mé databázi využívány ke stahování a automatickou aktualizaci dat ze služby Firebase. Tyto proměnné jsou zdrojem mých dat, které zobrazuji v šabloně způsobem `<ListView [items]="allevents$ async">`. Všechny události, které chceme zobrazit se načítají do proměnné *allevents$*. Pomocí *items* si můžeme všechny výsledky ukládat do proměnné *items*.

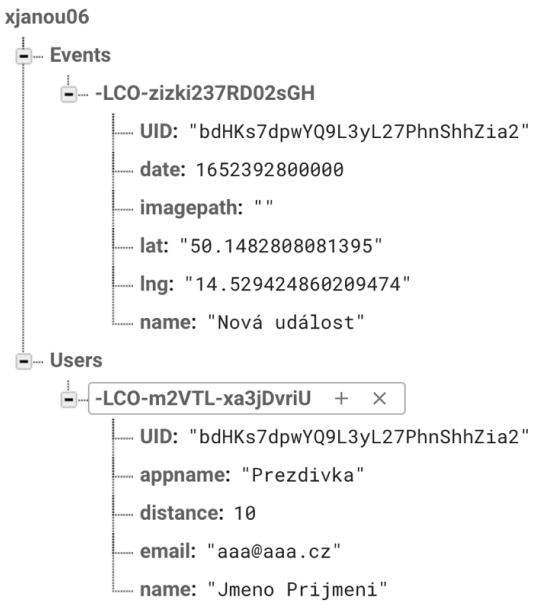
Díky vlastnosti Angularu *AsyncPipe* nám umožní přihlásit se k odběru Observe proměnné, když jsou komponenty inicializovány. Když je komponenta odstraněna, odhlásí se odběr. Data se následně předají do proměnné za pomoci:

```
<ng-template let-event="item">
<StackLayout (tap)="viewDetail(event.id)"></StackLayout>
<ng-template>
```

Proměnná byla definována pomocí *let-event* a nyní lze přistupovat k jejím hodnotám. Hodnoty v proměnné textit *allevents* jsou stahovány z real-time databáze Firebase.

7.2 Služba Firebase

Práci s touto databází provádíme pomocí pluginu NativeScriptu.[1] V momentě, kdy máme tento plugin pro NativeScript, přistupujeme přes jeho rozhraní k Real-time databázi, kterou jsme vytvořili a naplnili daty.



Obrázek 7.1: Naplnění databáze Firebase daty

[1] Plugin NativeScriptu `https://github.com/EddyVerbruggen/nativescript-plugin-firebase`

Za pomoci tohoto pluginu je možné využívat všechny aplikace, které Firebase nabízí. V této aplikaci využíváme také autentizaci pomocí e-mailu nebo Facebooku. Stahování dat z databáze se provádí přes servisní třídu *FirebaseService*, která je uvozena dekorátorem `@Injectable()`. Tato třída poskytuje služby, které se mohou využívat ve všech komponentách, kam ji importujeme. V této třídě pracujeme s databází a poskytuje tak metody pro vložení uživatelů do databáze nebo metody výběru dat z databáze.

```
addUser(user: User) {
    return firebase.push(
      "/Users",
      {"email": user.email,"name": user.name+" "+user.forename,
"appname": user.appname, "UID": BackendService.token, "distance":
user.distance}
    ).then(
      function(result: any) {
      },
      function(errorMessage: any) {
        console.log(errorMessage);
      });
  }
```

Obrázek 7.2: Metoda vložení dat uživatelů do databáze

Jednoduchým způsobem jsme předali data přes plugin Firebase, který nám poskytuje rozhraní pro vkládání dat metodou *push()*. Další metody, které lze použít jsou:

1. setValue()

```
// to store a JSON object
  firebase.setValue(
      '/companies',
      {foo:'bar'}
  );
```

2. getValue()

```
firebase.getValue('/companies')
      .then(result => console.log(JSON.stringify(result)))
      .catch(error => console.log("Error: " + error));
```

3. update()

```
firebase.update(
      '/companies',
      {'foo':'baz'}
  );
```

4. remove()

```
firebase.remove("/users");
```

7.3 Funkce používání pluginu MapBox

Využil jsem službu MapBox, která je poskytovatelem vlastních on-line map pro webové stránky a mobilní aplikace, díky které mohu zadávat v aplikaci body, kde se sportovní akce koná a zároveň ji vytvořit. MapBox v aplikaci mohu využívat díky pluginu pro NativeScript. Pro jeho používání je nezbytná registrace u této společnosti na jejich stránkách. Poté můžeme spravovat naší mapu, kterou lze různě editovat a nastavovat. Pro správnou funkci aplikace je zapotřebí vygenerovat token, pomocí kterého bude plugin přistupovat k našemu účtu a nastavení na stránkách MapBox.[2]

V této aplikaci je potom možné pomocí XML definice vložit mapu se stylem, který jsme si uložili a dalším nastavení definované přímo v XML. Proto je zde důležité nastavení *delay*, tedy zpoždění načtení mapy, aby bylo možné načtení dat z databáze a následně mapu zobrazit už s body jednotlivých událostí.

```
<Mapbox
accessToken="pk.eyJ1Ijoic3RvbmUxNCIsImEiOiJjamdkcHlzMzYycWdlMnpzMD
FlNjJ0ZDdhIn0.C96wzb2sqXyl4LOBAxdVig"

        mapStyle="moje"
        hideCompass="false"
        zoomLevel="15"
        delay="450"
        showUserLocation="true"
        disableZoom="false"
        disableRotation="false"
        disableScroll="true"
        disableTilt="true"
        (mapReady)="onMapReady($event)">
</Mapbox>
```

Nastaví se znemožnění pohybu mapy jedním prstem, protože v aplikaci přecházíme přetažením na další listy z jedné strany na druhou. Přidávání událostí na mapu probíhá touto posloupností:

1. Nejprve se načtou data z databáze pomocí pluginu Firebase
2. Data projdeme a zjistíme, která se budou vykreslovat
3. Pokud jsou události, které ještě neskončily a současně jsou ve vzdálenosti definované uživatelem, uloží se do pomocné struktury
4. Data se předají do mapy pomocí metody *addMarkes*

Pomocí metody *addMarkes* definujeme jejich polohu a další parametry. Parametrem je např. chování při stisknutí. V případě této aplikace se přesměrují na detail události.

[2] MapBox `https://www.mapbox.com`

```
Result = Definice jedné z událostí.
       this.map.addMarkers([
          {
            id: result.id,
            lat: result.lat,
            lng: result.lng,
            title: result.name,
            subtitle: 'Klikněte pro více informací',
            selected: false,
            onCalloutTap: () => {
              this.routerExtensions.navigate(["/list-detail-view", result.id],
{clearHistory: true});
            }
          }]);
```

7.4 Funkce geolokace za pomoci GPS

Součástí aplikace je funkce pro zjišťování polohy, kde se uživatel přesně v daný okamžik nachází. To se provádí přes modul geolokace. Aplikace umí zjistit naši současnou polohu přes GPS modul. Aplikace předává zeměpisné souřadnice *Longtitude* a *Latitude*, které se používají k jednoznačnému určení polohy na povrchu Země. Předáním těchto souřadnic do mapy, můžeme přesně definovat, kam dané body na mapě umístit.

```
// Get current location with high accuracy
geolocation.getCurrentLocation({ desiredAccuracy: Accuracy.high,
maximumAge: 5000, timeout: 20000 })
```

Kapitola 8

Testování

Testovací fáze celé mobilní aplikace je nedílnou a velice důležitou součástí celého vývoje. Cílem testování je odhalení chyb, které vznikly při vývoji aplikace nebo při samotném užívání aplikace reálnými uživateli. Odhalené chyby při testování pomáhají ke zlepšení kvality a funkčnosti celého výsledného produktu. Pokud by se testování neprovádělo kvalitně, jak při vývoji, tak při počátečním nasazení mezi reálné uživatele, vedlo by to ve výsledku ke značné ztrátě kvality. Ta by se projevila ze strany samotných uživatelů, ale hlavně při nahrávání a ověřování aplikace online distribuční službou.

Samotné testování probíhalo hlavně dvěma způsoby. Tím prvním bylo testováním za pomoci emulátoru při vyvíjení aplikace, kde bylo možné vidět a poznat základní nedostatky. Druhým způsobem testování byl za pomoci reálných uživatelů či testerů.

8.1 Testování za pomoci Emulátoru

Emulátor je druh softwaru, který slouží například k testování aplikací na různých platformách. Když vývojář vyvíjí aplikaci pro více platforem, ale vlastní a užívá pouze jednu z nich, použije emulátor, který mu pomůže zobrazit běh aplikace na těchto zařízeních virtuálním způsobem.

Velká nevýhoda těchto způsobů testování je nemožnost otestovat správnost činnosti geolokace využívající modul GPS, který používám v mé aplikaci nebo používání akcelerometru.

Xcode studio

Pro testování aplikace pro systém iOS je použito prostředí studia Xcode, díky kterému se snadno nahrála aktuální verze aplikace do mobilního telefonu značky Apple.

Android studio

Díky tomuto studiu bylo možné otestovat na mnoha typech mobilních zařízení virtuální cestou.

8.2 Testování aplikace pomocí TestFlight

Testování aplikace pomocí TestFlight bylo velice zajímavým řešením, a to proto, že jsem nemusel chodit na osobní schůzky s testery jako tomu bylo ve fázi Alfa testování. Stačilo jim na e-mail zaslat odkaz na testovací beta-verzi, kterou si nainstalovali do svých mobilních

telefonů pomocí aplikace TestFlight. Tato aplikace umožňuje zapojení až tisíce Apple ID, nikoliv samotných zařízení.[1]

Aby tento způsob testování mohl proběhnout, musel jsem mít developerský účet a přihlásit se na iTunes Connect, díky čemu se vytvořila betaverze pro testování a její šíření.[2]

Alfatestování

V první fázi uživatelského testování bylo prováděno tzv. **alfa testování**. To spočívá ve výběru jednoho potenciálního uživatele, který tuto aplikaci bude používat a při vývoji průběžně uživatelsky testovat. V praxi se mi tato metoda velmi osvědčila a byla mi velmi užitečnou. Samotné testování s jedním ochotným testerem, který nebyl odborníkem na informační technologie po celou dobu vývoje probíhala bez problémů. Touto metodou jsem, jako vývojář aplikace, dostával určitý druh zpětné vazby. Zpětnou vazbou byly informace, které poukázaly na vhodnost navržení uživatelského rozhraní a celého konceptu aplikace spolu s její hlavní myšlenkou.

Betatestování

Další metodou testování byla zvolena metoda **beta testování**, které spočívá v provedení testu uživatelem ve svém prostředí na svém mobilním zařízení. Výsledkem tohoto testování, je zpětná vazba určitého počtu uživatelů v podobě zprávy, která je zanesena do tabulky 8.1, která informuje na jakých zařízeních byla aplikace testována. V případě testování telefonů iPhone jsem musel chodit na předem sjednané schůzky, kde jsem aplikaci nahrával do telefonů díky developerskému účtu od společnosti Apple.

Výsledkem testování bylo příjemné zjištění ze strany testerů, kterým přišla tato myšlenka skvělá a vhodná pro další vývoj aplikace. Jeden z testerů mi řekl, že konečně nebude muset používat Facebook a zdlouhavě vytvářet sportovní tréninky, které pořádá výhradně pro své přátele. Důvodem bylo nepravidelné konání tréninků a také místa konání. Většinou se tréninky konaly pokaždé na jiném místě v závislosti na počasí a následné volbě typu tréninku. Tímto se tato aplikace stává užitečnou a vhodnou pro použití v reálném světě.

Alfa i beta testování jsem používal opakovaně. Důležité bylo minimalizovat možnost výskytu chyby či nevhodně navrhnutého řešení. Opravování při provozu je velice nepříjemné a velmi drahé. Může to mít např. dopad na míru použitelnosti célé mobilní aplikace.[9] newpage

Uživatelské beta testování				
č.	**Mobilní telefon**	**Typ OS**	**Verze OS**	**Poznámka testera**
1.	iPhone 5S	iOS	11.3.1	Žádný problém
2.	iPhone 6	iOS	11.3.1	Žádný problém
3.	iPhone X	iOS	11.3.1	Žádný problém
4.	iPhone 8 Plus	iOS	11.3.1	Žádný problém
5.	iPhone 5S	iOS	10.3.2	Žádný problém
(pokračování na další stránce)				

[1] TestFlight `https://itunesconnect.apple.com`

[2] `https://itunes.apple.com/cz/app/testflight/id899247664?l=cs&mt=8`

(pokracovani seznamu)				
č.	**Mobilní telefon**	**Typ OS**	**Verze OS**	**Poznámka testera**
6.	Huawei P9 Lite	Android	8.1 Oreo	Žádný problém
7.	HTC One	Android	6.0 Marshmallow	Žádný problém
(Konec seznamu)				

Tabulka 8.1: Uživatelské beta testování

Kapitola 9

Nasazení aplikace do distribuční služby

Závěrečnou fází vývoje mobilní aplikace je její distribuce k uživatelům. Tedy jaké možnosti má běžný potenciální uživatel, aby si mohl aplikaci jednoduchým způsobem stáhnout a nainstalovat do svého mobilního telefonu bez rizik poškození vlastního zařízení. Pro užité platformy existují dva bezpečné způsoby distribuce aplikace. Pro iPhone mobilní zařízení je to App Store a pro mobilní zařízení je to služba Google Play.

9.1 Apple Developer Program

Apple Developer Program je nezbytnou, velice důležitou částí při vývoji mobilní aplikace pro platformu iOS. Hlavní funkcí tohoto vývojářského programu je nahrání aplikace do distribuční aplikace AppStore společnosti Apple. Vytvoření členství vývojáře v základním programu je stanovena na 99$. Po založení a uhrazení částky získáme jednu z výhod, kterou je přístup k uživatelskému testování aplikace, kterou můžeme nahrát až na 100 mobilních zařízení od společnosti Apple včetně tabletů a chytrých hodinek. To značně zjednodušuje vývoj aplikace a následné testování. Důvodem pořízení tohoto programu, byla možnost snadného nahrání mobilní aplikace do mobilních zařízení uživatelů, kde byla aplikace testována metodou beta testování. Další podrobné informace a výhody o tomto programu nalezneme na internetových stránkách společnosti Apple.[1]

9.2 Google Play

Google Play služba společnosti Google umožňuje vývojářům nahrát mobilní aplikaci do distribuční sítě. V Google Play aplikaci je možné aplikaci zdarma stáhnout či za určitou cenu stanovenou vývojářem koupit. Než bude moci vývojář svoji aplikaci nahrát do této služby, musí se registrovat jako vývojář. U společnosti Google je registrační poplatek vývojářského programu stanoven na 25$. Aplikaci je možné šířit zdarma nebo za určitou cenu, která je uživatelem stanovena.

[1] Apple Developer Program `https://developer.apple.com`

Při publikování aplikace na Google Play je velice důležité rozmyslet si a zvážit, zda bude volně dostupnou nebo za určitý poplatek. Pokud se aplikace zveřejní prvotně ve formě volné aplikace, pak již nebude možné ji dodatečně publikovat za poplatek. Vývojář bude nucen vyvinout novou aplikaci. Pokud bude aplikace za poplatek 10$, tak si společnost Google nárokuje 30% z ceny aplikace. To znamená, že vývojáři náleží 7$ z vytvořené aplikace.[2]

[2]Služba Google Play `https://play.google.com/apps/publish/signup/`

Kapitola 10

Závěr

Cílem bakalářské práce bylo seznámit se s principy tvorby multi-platformních mobilních aplikací pro konkrétní platformy Android a iOS. Byla vyvinuta a implementována nativní mobilní aplikace, pro spojování lidí na sportovních akcích. Aplikace vznikla na popud přítele, který mne přivedl na základní myšlenku pro realizaci tohoto projektu. Hledal vhodnou a jednoduchou mobilní aplikaci, která bude zobrazovat akce sportovního charakteru v daném okolí. Mít možnost vyhledání určitého typu sportovní akce, připojení se a samotné vytváření sportovních akcí. Na základě této myšlenky jsem navrhl a implementoval mobilní aplikaci, která tyto základní požadavky splňuje.

Před samotným vývojem aplikace, jsem neprováděl velký průzkum podobných již existujících řešení. Samotným důvodem bylo, že tato první verze aplikace měla původně sloužit pouze pro široký okruh přátel a známých. Ve výsledku byla tato aplikace umístěna na příslušné distributivní místo společnosti Apple. Zveřejnění aplikace pro Android platformu zvažuji do budoucnosti. Testováním aplikace bylo zjištěno několik nedostatků, které byly záhy opraveny a opět testovány na malém vzorku dat, se kterým však bylo řešení aplikace důsledně konzultováno. Výsledkem tohoto projektu je plně funkční nativní mobilní aplikace pro obě platformy, která je dle mého názoru konkurenceschopná. Získal jsem velice cenné zkušenosti, znalosti a schopnost vyvíjet mobilní nativní aplikace způsobem zahrnující platformy iOS a Android. Zkušenosti, které jsem získal v oblasti návrhu aplikace, uživatelského rozhraní, implementace, testování, nasazení aplikace mezi samotné uživatele a do distribuční sítě jsou velmi cenné. Obzvláště, bylo zpočátku obtížné vyznat se v odvětví vývoje mobilních aplikací pro více platforem. Bylo a je mi velkým potěšením, pro mojí osobu, vytvořit projekt tohoto typu a představit ho běžným uživatelům, kteří se natolik nevyznají v oboru informačních technologií zabývající se vývojem mobilních aplikací. Mobilní aplikace bez problémů fungovala i na iPadu, což je typ zařízení typu tablet.

Budoucnost této aplikace vidím ve způsobu propagace mezi další uživatele širšího okruhu. Možné propojení s určitými rezervačními systémy, které používají určité fitness zařízení, pro rezervaci míst klientů daného sportu. Vytvoření seznamu přátel s možností posílání si zpráv mezi jednotlivými přáteli. Výhodou by bylo vytvoření notifikace, která by uživatelům zasílala zprávu o zrušení akce na mobilní telefon nebo e-mailovou adresu i potvrzení o účasti na sportovní akci. Dále možnost volby určitého jazyka, aby byla aplikace schopna použití mimo Českou republiku. Tím by uživatel měl jednodušší a lepší možnosti dostupnosti a informovanosti o sportech i v neznámém okolí, jako například nové město, které nezná dokonale.

Literatura

[1] AngularJS: *Programovací jazyk* . [online], Naposledy navštíveno 01.03.2018.
URL `https://cs.m.wikipedia.org/wiki/AngularJS`

[2] NativeScript: *Build amazing iOS and Android apps with technology you already know.* [online], Naposledy navštíveno 01.05.2018.
URL `https://docs.nativescript.org/start/how-it-works`

[3] OS Android: *Historie verzí Android* . [online], Naposledy navštíveno 04.02.2018.
URL `https://cs.wikipedia.org/wiki/Historie_verz%C3%AD_Android`

[4] NativeScript: *Build amazing iOS and Android apps with technology you already know.* [online], Naposledy navštíveno 08.05.2018.
URL `https://www.nativescript.org/`

[5] TypeScript: *Programovací jazyk* . [online], Naposledy navštíveno 10.05.2018.
URL `https://www.typescriptlang.org/`

[6] Model-view-controller: *Softwarová architektura* . [online], Naposledy navštíveno 15.02.2018.
URL `https://cs.wikipedia.org/wiki/Model-view-controller/`

[7] Google: *Dashboards.* [online], Naposledy navštíveno 28.03.2018.
URL `https://developer.android.com/about/dashboards/index.html#Platform`

[8] Brown, E.: *Learning JavaScript: JavaScript Essentials for Modern Application Development.* O'Reilly Media, Inc., třetí vydání, 2016, ISBN 1491914920.

[9] Křena, B.; Kočí, R.: *Úvod do softwarového inženýrství. IUS. Studijní opora.* 2010.

[10] Luboslav Lacko: *Vývoj aplikací pro Android.* Computer Press Brno, 2015, ISBN 978-80-251-4347-6.

[11] Lundrigan, L.; Allen, S.; Graupera, V.: *Pro Smartphone Cross–Platform Development.* Apress L. P., 2010, ISBN 9781430228684.

[12] Occhino, T.: React Native: *Bringing modern web techniques to mobile.* [online], Naposledy navštíveno 06.04.2018.
URL `https://code.facebook.com/posts/1014532261909640/react-native-bringing-modern-web-techniques-to-mobile/`

[13] Ujbányai, M.: *Programujeme pro Android.* Grada Publishing, 2012, ISBN 978-80-247-3995-3.

[14] Vávrů, J.: *iPhone: vývoj aplikací*. Grada Publishing, 2012, ISBN 978-80-247-4457-5.

[15] Walker, N.; Anderson, N. J.: *NativeScript: NativeScript for Angular Mobile Development*. Packt Publishing Ltd, 2017, ISBN 9781787124691.

Přílohy

Příloha A

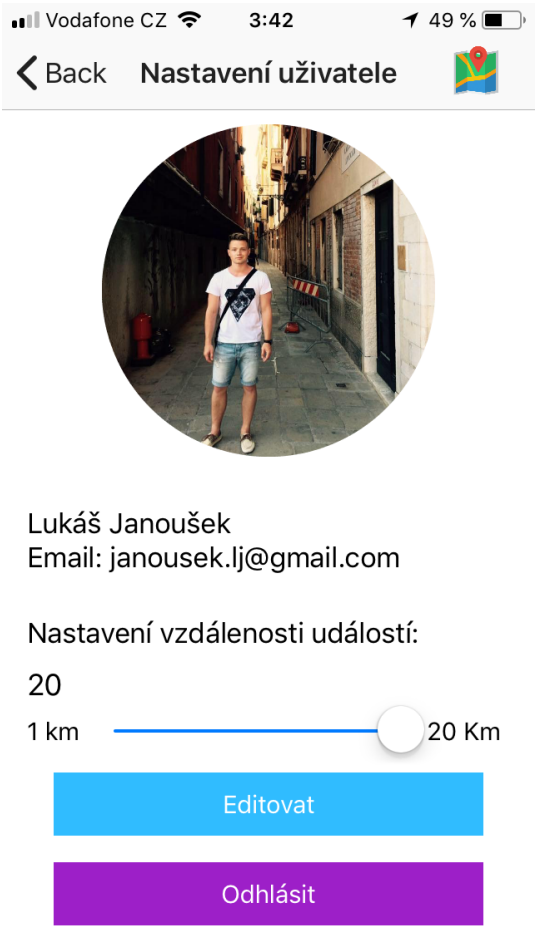
Obsah přiloženého paměťového média

Přiložené paměťové DVD má následující adresářovou strukturu:

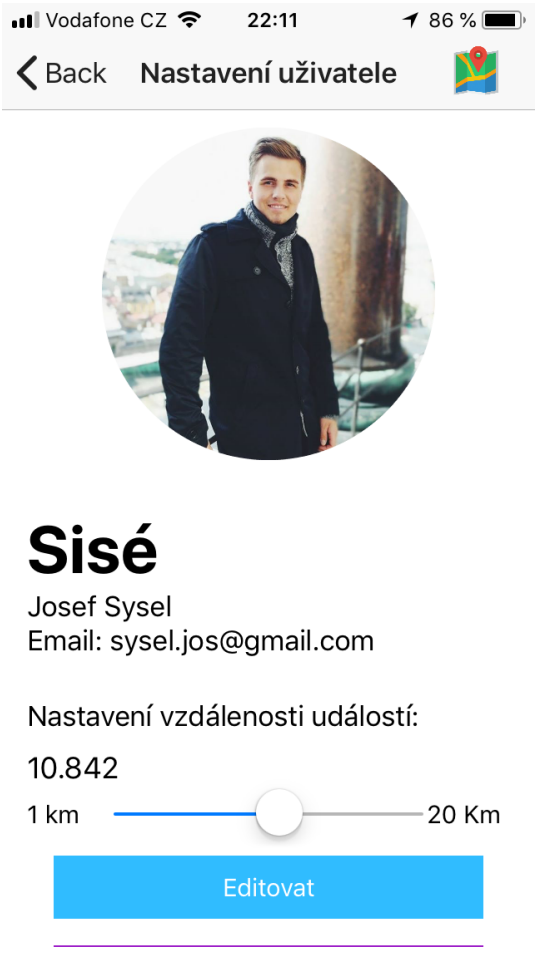
- **Zdrojové soubory**
  - Projekt.zip
- **Dokumenty**
  - BP.pdf
  - Zdrojové kódy LaTeX
  - README.txt

Příloha B

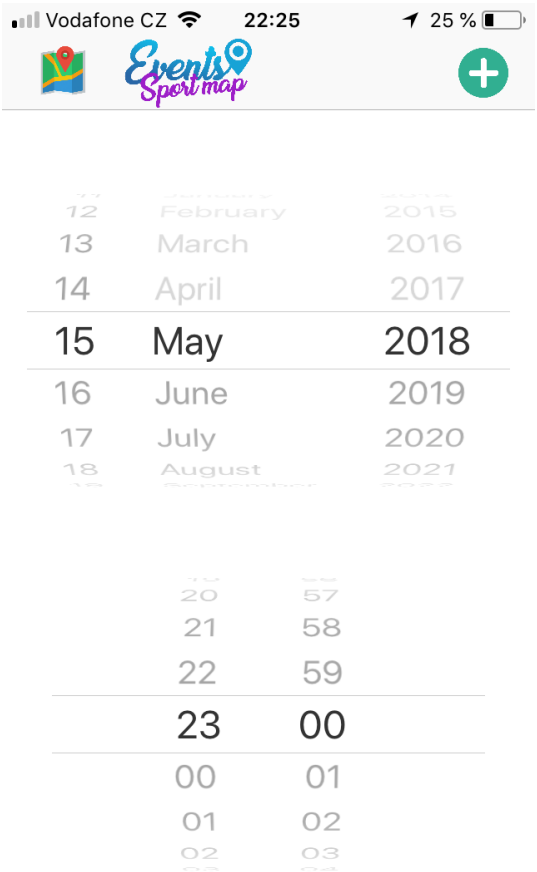
Výsledný vzhled mobilní aplikace



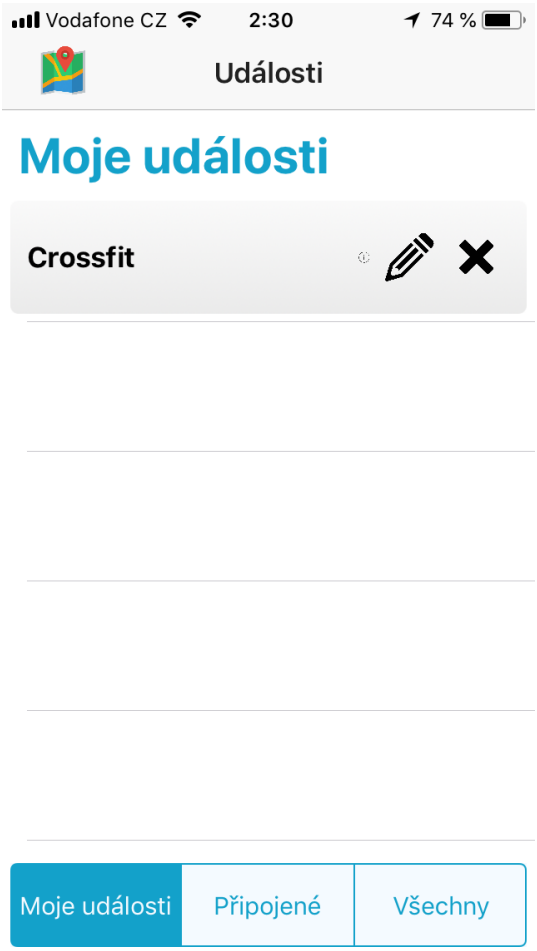
Obrázek B.1: Nastavení uživatele



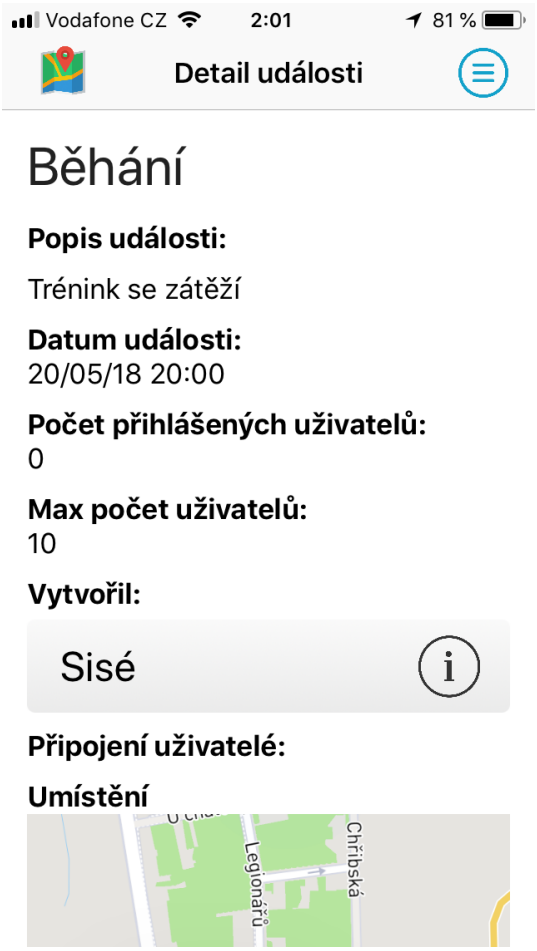
Obrázek B.2: Nastavení profilu testera



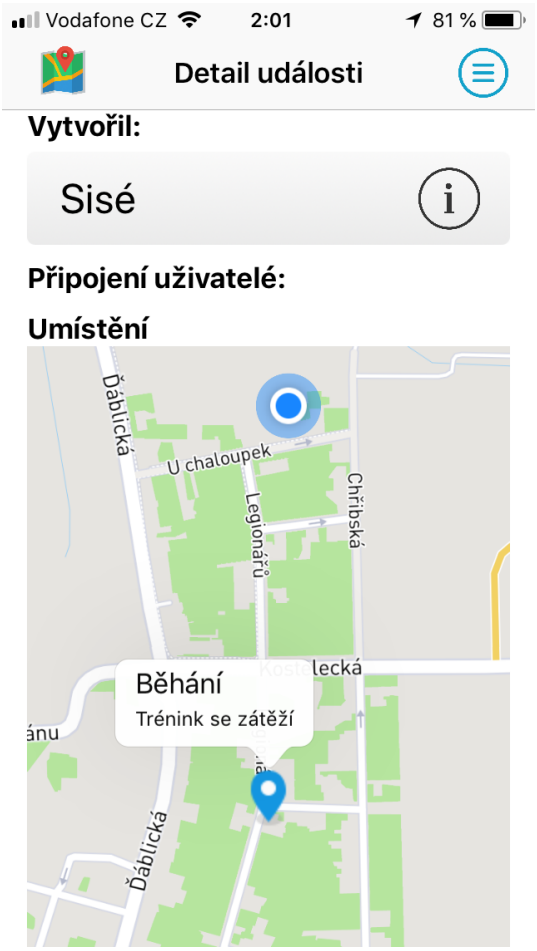
Obrázek B.3: Určení data a času události



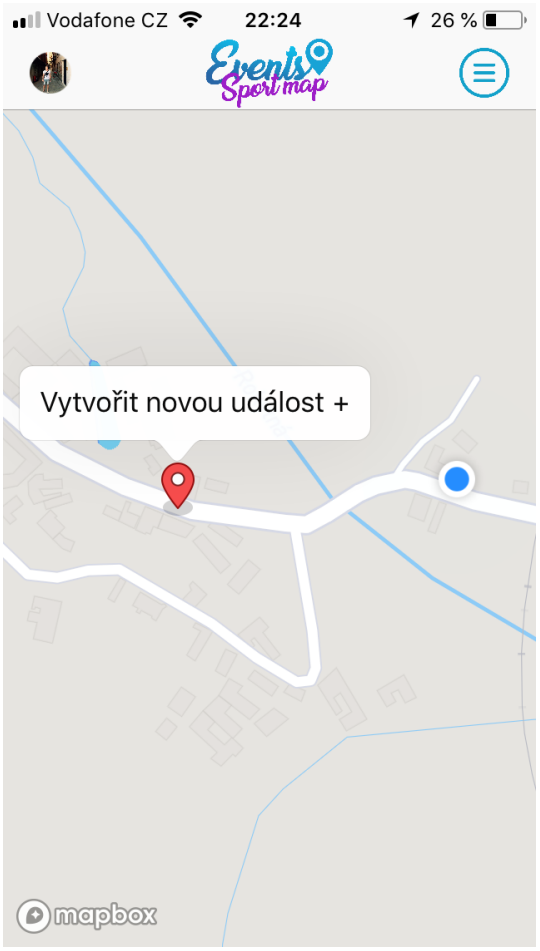
Obrázek B.4: Moje vytvořené události



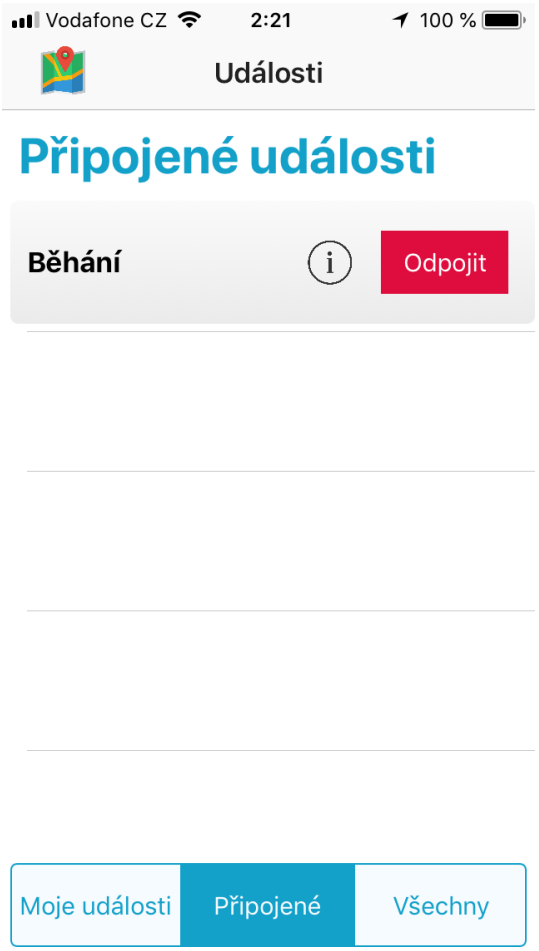
Obrázek B.5: Vytvořená událost-část 1



Obrázek B.6: Vytvořená událost-část 2



Obrázek B.7: Bod vytvoření události



Obrázek B.8: Detail připojených událostí